# Algorithms for the Clique Problem with Multiple-Choice Constraints under a Series-Parallel Dependency Graph

Andreas Bärmann [1], Patrick Gemander [2], Maximilian Merkert [3], Ann-Kathrin Wiertz [4], Francisco Javier Zaragoza Martínez [5]

**Abstract**

The *clique problem with multiple-choice constraints* (CPMC), i.e. the problem of finding a $k$-clique in a $k$-partite graph with known partition, occurs as a substructure in many real-world applications, in particular scheduling and railway timetabling. Although CPMC is NP-complete in general, it is known to be solvable in polynomial time when the so-called *dependency graph* of $G$ is a forest. In this article, we focus on the special case CPMCSP, where the dependency graph of $G$ is series-parallel. We give a polynomial-time algorithm for CPMCSP using dynamic programming. Further, we provide some facet-inducing inequalities of the CPMCSP polytope, mainly using properties of the stable set polytope of the complement graph of $G$. Among these, we give a separation algorithm for the so-called *embedded odd-clique-cycle* inequalities using dynamic programming. If the number of vertices per subset of the $k$-partition is bounded, then its runtime is polynomial in the size of the dependency graph.

**Keywords:** Clique Problem, Dynamic Programming, Integer Programming, Multiple-Choice Constraints, Series-Parallel Graphs

**Mathematics Subject Classification (MSC2020):** 90C10, 90C27, 90C39, 90C57, 05C69, 05C38, 05C85

[1] *Friedrich-Alexander-Universität Erlangen-Nürnberg, Department of Data Science / Department Mathematik, Cauerstraße 11, 91058 Erlangen, Germany,* andreas.baermann@fau.de

[2] *Fraunhofer-Institut für Integrierte Schaltungen IIS, Gruppe Optimization, Nordostpark 93, 90411 Nürnberg, Germany,* patrick.gemander@fau.de

[3] *Technische Universität Braunschweig, Institute for Mathematical Optimization, Universitätsplatz 2, 38106 Braunschweig, Germany,* m.merkert@tu-braunschweig.de

[4] *Friedrich-Alexander-Universität Erlangen-Nürnberg, School of Business, Economics and Society, Lange Gasse 20, 90403 Nürnberg, Germany,* ann-kathrin.wiertz@fau.de

[5] *Universidad Autónoma Metropolitana Azcapotzalco, Departamento de Sistemas, Av. San Pablo 180, 02200 Ciudad de México, Mexico,* franz@azc.uam.mx

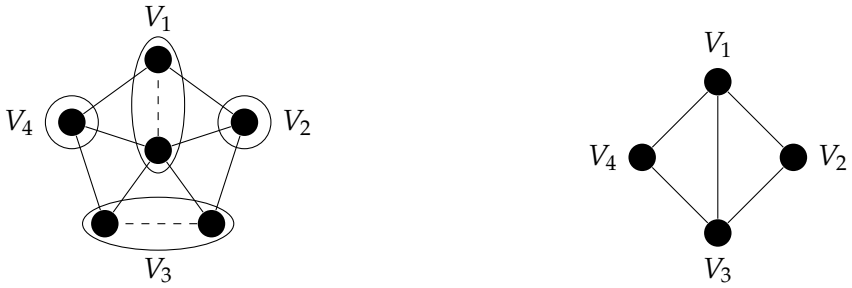
# 1 Introduction

A prevalent substructure in many real-world applications, especially those containing scheduling aspects, is the *clique problem with multiple-choice constraints* (CPMC). It has been shown to be NP-complete via reduction to $k$-colouring, see [10]. The problem consists in finding a $k$-clique in a $k$-partite graph $G$ with a given $k$-partition $\mathcal{V} = \{V_1, \dots, V_k\}$ of its vertex set $V(G)$. Hence, an instance of CPMC is described by the pair $(G, \mathcal{V})$. We call $G$ the *compatibility graph* of the CPMC instance, while the complement graph $\overline{G}$ of $G$ is called the *conflict graph*.

For the analysis of CPMC instances, we use an induced structure representing dependencies between choices made in distinct subsets $V_i, V_j \in \mathcal{V}$. We denote by $G_{ij}$ the (bipartite) subgraph of $G$ induced by $V_i \cup V_j$. The instance of CPMC then induces the *dependency graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where an edge $\{V_i, V_j\}$ is contained in $\mathcal{E}$ if and only if the corresponding $G_{ij}$ is complete bipartite, i.e. each vertex in $V_i$ is connected to all vertices in $V_j$. In terms of conflict graphs, an edge $\{V_i, V_j\}$ is contained in $\mathcal{E}$ if and only if the corresponding $\overline{G}_{ij}$ has no edges between $V_i$ and $V_j$.

The underlying idea is that for a clique in $G$, two vertices $u \in V_i$ and $v \in V_j$ can be chosen independently of each other if and only if the edge $\{V_i, V_j\}$ is not contained in $\mathcal{E}$. Consequently, partial solutions of CPMC corresponding to different connected components of the dependency graph are always compatible. CPMC can thus be solved independently on each connected component of the dependency graph. Therefore, we only consider CPMC instances with a connected dependency graph in the remainder of this work.

As an illustration of the above definitions, consider the conflict graph $\overline{G}$ with dependency graph $\mathcal{G}$ shown in Figures 1a and 1b respectively.




(a) Example conflict graph $\overline{G}$ (b) Dependency graph $\mathcal{G}$

Figure 1: Example instance of CPMC. Dashed lines represent edges within subsets $V_i \in \mathcal{V}$.

In our illustrations, we connect vertices belonging to the same subset $V_i \in \mathcal{V}$ with dashed lines. The partition of this conflict graph consists of the four subsets $\mathcal{V}_1, \dots, \mathcal{V}_4$. The corresponding dependency graph then contains all edges except for $\{V_2, V_4\}$, as there is no edge between nodes from $V_2$ and $V_4$, while all other subgraphs $\overline{G}_{ij}$ have connections between $V_i$ and $V_j$. In this example, the CPMC instance has no solution.

To the best of our knowledge, the dedicated study of polyhedral properties of CPMC started in [7] and has been continued in [10]. These works examine two polynomial-time solvable special cases of CPMC and give complete polyhedral descriptions for both. The latter considers CPMC instances with cycle-free dependency graphs, or in other words, dependency graphs with tree-width one. In the present article, we consider CPMC under a series-parallel dependency graph (CPMCSP), which means it has tree-width two.

**Contribution** We begin our study by generalizing the dynamic program used in [10] to solve CPMC if the dependency graph is a forest to also solve CPMCSP in polynomial time. Using a result from [25], this also implies the existence of an extended formulation of polynomial size. Next, we outline some connections between CPMC and stable set theory. In particular, the well-known result that the support graph of a facet-defining inequality of the stable set polytope

must be 2-connected also holds for the dependency graph of the support graph. Since the CPMC polytope can be interpreted as a face of the stable set polytope, this is a welcome result when it comes to separation techniques. Further, we state two simple reduction techniques for instances $(G, \mathcal{V})$ of CPMC. If, after application of these reductions, the dependency graph is a hole and all subsets in the partition $\mathcal{V}$ are of size two, the complement graph $\overline{G}$ of $G$ turns out to be $h$-perfect and, therefore, the complete polyhedral description is given by the odd-cycle constraints, clique constraints and non-negativity constraints. When allowing larger subsets in $\mathcal{V}$, we generally lose $h$-perfectness of $\overline{G}$, but we discovered that the odd-cycle constraints can be generalized. Namely, instead of vertices $v_1, \dots, v_{\ell+1} = v_1$ forming a cycle in $\overline{G}$ we consider sets of vertices $C_1, \dots, C_{\ell+1} = C_1$ such that the unions $C_n \cup C_{n+1}$ form cliques, and hence we get a *clique cycle* in $\overline{G}$. A major contribution then is a separation algorithm for clique-cycle constraints whose clique cycle $\mathcal{C} = \{C_1, \dots, C_\ell\}$ is of odd length $\ell$ and embedded in the partition $\mathcal{V}$, i.e. for each clique $C$ in $\mathcal{C}$ there exists some subset $V_i$ in $\mathcal{V}$ with $C \subseteq V_i$.

**Related Literature** As indicated, prior polynomial special cases of CPMC have been examined in the literature. The authors of [7, 9] study CPMC under so-called *staircase compatibility*, where the CPMC polytope can be fully described via a totally unimodular linear constraint system of polynomial size. In contrast to staircase compatibility, which imposes a certain local criterion on the subgraphs $G_{ij}$, the polynomial special case studied in [10] requires a global criterion to be fulfilled: the dependency graph $\mathcal{G}$ needs to be a forest. Here, the CPMC polytope can be fully described by stable set constraints (and the non-negativity bounds) as the compatibility graph $G$ turns out to be perfect.

CPMC is an important structure in many real-world problem contexts, where it can be used to model the compatibility of choices to be made from certain subsets of options. A typical example is choosing an admissible time slot for given tasks that have to be scheduled. Thus, CPMC is relevant for a wide variety of scheduling applications in which feasibility of a solution can be described by pairwise compatibilities of individual choices. The concept is particularly useful for scheduling problems with precedence constraints, such as the project scheduling problem (see [28] for a broad overview). Practical applications of CPMC to railway timetabling are given in [7, 10, 12, 8, 11]. CPMC also occurs as a key structure in locomotive scheduling and driver rostering, see [13]. The authors of [7, 23] study it in the context of piecewise linear flows.

A $k$-clique in $G$ is also equivalent to an *independent transversal* or an *independent system of representatives* in the complement graph $\overline{G}$. In [22], a criterion is proved to guarantee the existence of an independent transversal, and in [1] the authors take on a colourability conjecture. Another framework that fits this setting are the *systems of disjoint representatives* for a family of hypergraphs, where a hypergraph version of Hall's theorem is proved in [2]. The authors of [20] consider the enumeration of all $k$-cliques in a $k$-partite graph in the context of textile engineering, and a subsequent improvement for their enumeration method is proposed in [26].

For an introduction to the general clique problem and its polyhedral properties, we refer to [19, 6]. A broad survey on algorithms for the maximum clique problem can be found in [5].

**Structure of the Paper** Some general notation is introduced in Section 2, where we also restate relevant results from the literature on series-parallel graphs. The polynomial-time solvability of CPMCSP is proved in Section 3 via a dynamic program. In Section 4, we restate relevant polyhedral results from the literature and then show that support dependency graphs of facet-defining inequalities of the stable set polytope must be 2-connected. Section 5 opens with two simple reduction techniques and then deals with the particular case where the dependency graph is a hole and all subsets in $\mathcal{V}$ are of size two. In Section 6, we generalize odd-cycle constraints to odd-clique-cycle constraints and give a separation algorithm for a subset of them, which we call *embedded clique-cycle constraints*. We end with our conclusions in Section 7.

## 2 Notation and Preliminaries on Series-Parallel Graphs

In this section, we introduce some general notation, some of which is summarized in Table 1. We also restate some basic results for series-parallel graphs.

| Symbol | Meaning |
|---|---|
| $G$ | graph (mostly a compatibility graph on which we consider clique problems) |
| $\overline{G}$ | complement graph of $G$ (mostly a conflict graph on which we consider stable set problems) |
| $V(G), E(G)$ | vertex set and edge set of $G$ |
| $N_G(v)$ | neighbourhood of vertex $v$ in $G$ |
| $\mathcal{V}$ | partition of vertex set $V(G)$ |
| $V_i, V_j$ | subsets in the partition $\mathcal{V}$ |
| $G_{ij}$ | subgraph induced by two distinct subsets $V_i, V_j \in \mathcal{V}$ |
| $C$ | clique in a graph |
| $\mathcal{C}$ | set of cliques (typically forming a clique path or clique cycle; see Section 6) |
| $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ | dependency graph $\mathcal{G}$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E} := \{\{V_i, V_j\} \subseteq \mathcal{V} \colon G_{ij} \text{ is not complete bipartite}\}$ |
| $\mathcal{T}$ | decomposition tree of a series-parallel dependency graph $\mathcal{G}$ (Definition 2.2) |
| $G[a]$ ($\mathcal{G}[a]$) | support (dependency) graph of inequality $a^\top x \leq \alpha$ (Definition 4.3) |
| $k$ | number of subsets in the partition $\mathcal{V}$ |
| $\ell$ | length of paths or cycles |
| $u, v, w$ | vertices in some compatibility or conflict graph |
| $e$ | node of a decomposition tree $\mathcal{T}$ |
| $s, t$ | indices for terminal-related notation, e.g. for terminals $V_s, V_t \in \mathcal{V}$ of a series-parallel dependency graph $\mathcal{G}$ |

Table 1: Notation overview

Unless specified otherwise, we consider all occurring graphs to be simple and undirected. For a graph $G$, we denote by $V(G)$ and $E(G)$ the vertex set and the edge set of $G$, respectively. The neighbourhood of a vertex $v \in V(G)$ is denoted by $N_G(v) := \{u \in V(G) : \{u, v\} \in E(V)\}$. A set $W \subseteq V(G)$ is called a *clique* (resp. *stable set*) in $G$, if for any two vertices $u, v \in W$ the edge $\{u, v\}$ is contained (resp. not contained) in $E(G)$. Further, we call a sequence of pairwise distinct vertices $(v_1, \ldots, v_{\ell+1})$ a *path* of length $\ell$ in $G$ if $\{v_i, v_{i+1}\} \in E(G)$ for $i \in \{1, \ldots, \ell\}$. A *cycle* of length $\ell$ in $G$ is a path $(v_1, \ldots, v_\ell)$ where the edge $\{v_\ell, v_1\}$ is also contained in $E(G)$. Edges between vertices of the cycle that are not of the form $\{v_i, v_{i+1}\}$ (where indices are taken modulo $\ell$) are called *chords*. A cycle that has no chord in $G$ is called a *hole*.

Later on, we discuss CPMC instances with series-parallel dependency graphs. There are several – not necessarily equivalent – definitions of series-parallel graphs. In this work, we make use of a version introduced by Wimer and Hedetniemi as a class contained in the family of $k$-terminal recursive graphs [30].

**Definition 2.1** (Generalized 2-terminal series-parallel graphs (GSP) [21])**.** *GSP graphs are exactly the triples $(G, s, t)$ formed by a graph $G$ and two distinct terminals $s, t \in V(G)$ defined recursively as follows:*

**Base case** *If $V(G) = \{s, t\}$ and $E(G) = \{\{s, t\}\}$, then $(G, s, t)$ is a GSP graph.*

**Series composition 'S'** *If $(G_1, s_1, t_1)$ and $(G_2, s_2, t_2)$ are GSP graphs, then $(G, s_1, t_2)$ is a GSP graph, where $G$ is the graph obtained from $G_1$ and $G_2$ by identifying $t_1$ with $s_2$.*

**Parallel composition 'P'** *If $(G_1, s_1, t_1)$ and $(G_2, s_2, t_2)$ are GSP graphs, then $(G, s_1, t_1)$ is a GSP graph, where $G$ is the graph obtained from $G_1$ and $G_2$ by identifying $s_1$ with $s_2$ and $t_1$ with $t_2$.*

**Generalized series composition 'G'** *If $(G_1, s_1, t_1)$ and $(G_2, s_2, t_2)$ are GSP graphs, then $(G, s_2, t_2)$ is a GSP graph, where $G$ is the graph obtained from $G_1$ and $G_2$ by identifying $t_1$ with $s_2$.*

At this point we remark that the class of GSP graphs properly contains the more well-known class of *2-terminal series-parallel graphs* (2SP), which are obtained by using only the base case

and the series and parallel compositions above. Also note that all GSP graphs, as well as all 2SP graphs, are connected.

We also make use of the decomposition tree that defines a particular GSP graph.

**Definition 2.2** (Decomposition tree)**.** *A* decomposition tree *$T$ of a GSP graph $(G, s, t)$ is the binary tree given by:*

**Leaf case** *If $G$ consists of exactly one edge $e$, then $T$ consists of exactly one node labelled $e = (s, t)$.*

**Composition case** *If $G$ results from some composition of two GSP graphs $(G_1, s_1, t_1)$ and $(G_2, s_2, t_2)$ with decomposition trees $T_1$ and $T_2$, respectively, then $T$ consists of a root $r$ labelled by the corresponding composition (either 'S', 'P', or 'G'), with left subtree $T_1$ and right subtree $T_2$.*

Those graphs $G$ for which $(G, s, t)$ is a GSP graph for some $s, t \in V(G)$ will from now on simply be called *series-parallel* graphs. This choice is justified by the following equivalence of alternative definitions of series-parallel graphs.

**Theorem 2.3** (Series-parallel graphs)**.** *Let $G$ be a connected graph. The following are equivalent:*

1. *$G$ has no $K_4$ minor.*
2. *$G$ has no $K_4$ subdivision.*
3. *$G$ has treewidth at most 2.*
4. *$(G, s, t)$ is a GSP graph for some $s, t \in V(G)$.*
5. *Each 2-connected component of $G$ is a 2SP graph.*

The proof of these equivalences is distributed over Duffin [17], Bodlaender [3], Wald and Colbourn [29], Bodlaender and van Antwerpen [4] as well as Wimer and Hedetniemi [30].

# 3 A Dynamic Programming Algorithm

Next, we prove that CPMCSP is solvable in polynomial time by a dynamic programming algorithm.

**Theorem 3.1.** *CPMCSP is solvable in $\mathcal{O}(|\mathcal{V}| \cdot |V_{\max}|^3)$ time, where $V_{\max} \in \mathcal{V}$ denotes a subset of maximum cardinality.*

*Proof.* We provide a dynamic programming algorithm to solve the weighted CPMCSP on a given $k$-partite compatibility graph $G$ with partition $\mathcal{V}$ and costs $c_v \in \mathbb{R}$ for $v \in V(G)$.

Let $\mathcal{T}$ be a decomposition tree of the dependency graph $\mathcal{G}$. Further, let $e \in V(\mathcal{T})$ be a leaf of the decomposition tree with terminals $V_s, V_t \in \mathcal{V}$. In the following, we describe how the leaf can be processed if it represents either a graph consisting of only the edge $\{V_s, V_t\}$ or a composition of two subgraphs. More precisely, we calculate for all $(u, v) \in V_s \times V_t$ the minimal weight $f_e(u, v)$ of any clique $C$ in the subgraph of $G$ represented by $e$ with $\{u, v\} \subseteq C$. If no such clique exists, the minimal weight is set to infinity. Note that this can only occur due to subgraphs $G_{ij}$ with $\{V_i, V_j\} \in E(\mathcal{G})$, as all other subgraphs $G_{ij}$ are complete bipartite. Once a leaf $e$ is processed, it is removed from the decomposition tree $\mathcal{T}$.

**Base case** If $e$ is a leaf representing a graph consisting of only the edge $\{V_s, V_t\}$, then we define for $u \in V_s$ and $v \in V_t$

$$f_e(u, v) := \begin{cases} c_u + c_v, & \text{if } \{u, v\} \in E(G) \\ \infty, & \text{otherwise.} \end{cases} \tag{1}$$

**Series composition 'S'** If $e$ represents a series composition of two children $e_1, e_2$ with terminals $V_{s_1}, V_{t_1}$ and $V_{s_2}, V_{t_2}$ respectively, then we have $V_{t_1} = V_{s_2}$, and we define for $u \in V_s = V_{s_1}, v \in V_t = V_{t_2}$

$$f_e(u,v) \coloneqq \min_{w \in V_{t_1}} f_{e_1}(u,w) + f_{e_2}(w,v) - c_w. \tag{2}$$

**Parallel composition 'P'** If $e$ is a parallel composition of two children $e_1, e_2$ with terminals $V_{s_1}, V_{t_1}$ and $V_{s_2}, V_{t_2}$ respectively, then we have $V_{s_1} = V_{s_2}, V_{t_1} = V_{t_2}$, and we define for $u \in V_s = V_{s_1}, v \in V_t = V_{t_1}$

$$f_e(u,v) \coloneqq f_{e_1}(u,v) + f_{e_2}(u,v) - c_u - c_v. \tag{3}$$

**Generalized series composition 'G'** Finally, if $e$ is a generalized series composition of two children $e_1, e_2$ with terminals $V_{s_1}, V_{t_1}$ and $V_{s_2}, V_{t_2}$ respectively, then we have $V_{t_1} = V_{s_2}$ and we define for all $u \in V_s = V_{s_2}$ and $v \in V_t = V_{t_2}$

$$f_e(u,v) \coloneqq \min_{w \in V_{s_1}} f_{e_1}(w,u) + f_{e_2}(u,v) - c_u. \tag{4}$$

The globally optimal solution of CPMCSP on the graph $G$ can be computed as

$$\min_{u \in V_s, v \in V_t} f_r(u,v), \tag{5}$$

where $r \in V(\mathcal{T})$ is the root of $\mathcal{T}$ with terminals $V_s, V_t \in \mathcal{V}$.

A decomposition tree $\mathcal{T}$ can be computed in $\mathcal{O}(|\mathcal{V}|)$ time, see [30, Theorem 3]. Processing a leaf $e \in V(\mathcal{T})$ representing an edge $\{V_s, V_t\}$ requires $\mathcal{O}(|V_{\max}|^2)$ time, which is also the maximum time requirement for parallel compositions. A series or generalized series composition takes at most $\mathcal{O}(|V_{\max}|^3)$ time. Since the size of $\mathcal{T}$ is bounded by $\mathcal{O}(|\mathcal{V}|)$, we get a total time complexity of $\mathcal{O}(|\mathcal{V}| \cdot |V_{\max}|^3)$. □

The resulting dynamic programming algorithm to solve CPMCSP is summarized as Algorithm 1, which also includes the construction of an optimal clique if it exists.

From a polynomial-time dynamic programming algorithm for a combinatorial optimization problem, it is often possible to derive an extended formulation of polynomial size for that problem. This result is due to [25]. Their construction translates the possible decisions during the dynamic program into a hyperflow in directed hypergraphs, so-called *decision hypergraphs*. For Algorithm 1 the decision hypergraph $\mathcal{H}$ has several stages, corresponding to nodes of the decomposition tree $\mathcal{T}$. In each stage $e$, we have a vertex for every pair of vertices $(u,v) \in V_s \times V_t$, where $V_s, V_t$ are the terminals of $e$. This means we have a vertex for every partial solution element $f_e(u,v)$ the dynamic programming algorithms keeps track of. Directed hyperarcs $(J,l)$ of $\mathcal{H}$ correspond to decisions of the dynamic programming algorithm. There exists such a hyperarc whenever several partial solutions corresponding to nodes in the set $J$ may be passed on in one of the recursions for computing $l$, i.e. in the recursions (2), (3), (4) for any of the cases 'S', 'P', 'G', or in (5) for the global solution element. We therefore observe that in our case hyperarcs consist of at most three elements.

As shown in [25, Cor. 1], the problem formulation as a hyperflow in $\mathcal{H}$ has the *total dual integrality* property if the decision hypergraph $\mathcal{H}$ is acyclic. This property is fulfilled in our case as all vertices are naturally ordered by their level in the decomposition tree, such that the order of elements from $J$ is always smaller than the order of $l$, as required. Moreover, the extreme points of the underlying polyhedron correspond exactly to the outcome of a dynamic programming run if a further requirement is fulfilled: the existence of a *reference subset* with *consistency* and *disjointness* properties – as they are called in [25]. This essentially means that every partial solution element $f_e(u,v)$ can contribute to the global solution element defined

**Algorithm 1:** Dynamic programming algorithm for CPMCSP with linear vertex costs

**Input:** CPMCSP-instance $(G, \mathcal{V})$ with connected dependency graph $\mathcal{G}$ and costs $c_v \in \mathbb{R}$ for $v \in V(G)$

**Output:** A weight-minimal clique $C$ in $G$ satisfying the multiple-choice constraints

Compute a decomposition tree $\mathcal{T}$ of $\mathcal{G}$
**while** $V(\mathcal{T}) \neq \emptyset$ **do**
  Select $e \in V(\mathcal{T})$ with $\deg_{\mathcal{T}}(e) = 1$
  **foreach** $(u, v) \in V_s \times V_t$ **do**
    **if** ***Base case*** **then**
      Calculate $f_e(u, v)$ using (1)
      $C_{e,u,v} \leftarrow \{u, v\}$
    **else if** ***Series composition*** **(*S*) then**
      Calculate $f_e(u, v)$ using (2)
      Find $w \in V_{t_1}$ with $f_{e_1}(u, w) + f_{e_2}(w, v) - c_w = f_e(u, v)$
      $C_{e,u,v} \leftarrow C_{e_1,u,w} \cup C_{e_2,w,v}$
    **else if** ***Parallel composition*** **(*P*) then**
      Calculate $f_e(u, v)$ using (3)
      $C_{e,u,v} \leftarrow C_{e_1,u,v} \cup C_{e_2,u,v}$
    **else if** ***Generalized series composition*** **(*G*) then**
      Calculate $f_e(u, v)$ using (4)
      Find $w \in V_{s_1}$ with $f_{e_1}(w, u) + f_{e_2}(u, v) - c_u = f_e(u, v)$
      $C_{e,u,v} \leftarrow C_{e_1,w,u} \cup C_{e_2,u,v}$
  $r \leftarrow e$
  Remove $e$ from $V(\mathcal{T})$
Select $(u, v) \in V_s \times V_t$ with $f_r(u, v) = \min_{(u',v') \in V_s \times V_t} f_r(u', v')$
**if** $f_r(u, v) < \infty$ **then**
  **return** $C_{r,u,v}$
**else**
  **return** 'Problem has no solution'

by (5) at most once during the algorithm. This requirement is also fulfilled due to $\mathcal{T}$ being a tree: indeed, every solution element $f_e(u, v)$ can contribute to (5) (directly or indirectly) only through the unique path from $e$ leading up to the root. Thus, the construction from [25] can be applied to Algorithm 1.

In fact, a subsection of [25] is especially devoted to recursively defined graph families as they often allow for dynamic programming algorithms with the required properties. In the same article, the interested reader will also find an example on a Steiner tree algorithm on series-parallel graphs from the literature. Further examples can also be found in the survey [15].

Since the linear programming formulation resulting from Algorithm 1 will have a variable for every hyperedge and a constraint for each node of $\mathcal{H}$, it follows from the proof of Theorem 3.1 that the number of variables and constraints is in $\mathcal{O}(|\mathcal{V}| \cdot |V_{\max}|^3)$.

# 4 Basic Polyhedral Aspects

We now restate important results on the stable set polytope from the literature in Section 4.1 and discuss the connectedness of support dependency graphs in Section 4.2.

## 4.1 Results from the Literature

The *stable set polytope* (resp. *clique polytope*) of a graph $G$ is formally defined as the convex hull of all incidence vectors of stable sets (resp. cliques) in $G$. Later on, we will relate to two prominent classes of graphs where the complete description of the stable set polytope is known. Namely, as stated in [19], a graph $G$ is *perfect* if the inequalities

$$\sum_{v \in C} x_v \leq 1 \qquad \forall \text{ cliques } C \text{ in } G$$
$$x_v \geq 0 \qquad \forall v \in V(G)$$

describe its stable set polytope, and the graph is *h-perfect* if its stable set polytope is described by the inequalities above together with

$$\sum_{v \in Q} x_v \leq \frac{|Q|-1}{2} \qquad \forall \text{ odd cycles } Q \text{ in } G.$$

In relation to the stable set polytope, we will later use the following two important results by Chvátal from [14].

**Theorem 4.1** (Node substitution [14])**.** *Let $H'$ and $H''$ be two graphs and let $A'x' \leq b'$ and $A''x'' \leq b''$ be complete descriptions of their respective stable set polytopes. Further, let $v \in V(H')$, and let $H$ be the graph resulting from substitution of $v$ by $H''$, i.e. replacing $v$ in $H'$ by $H''$ and connecting each vertex in $V(H'')$ to each vertex in $N_{H'}(v)$. Then the system*

$$a'_{iv} \sum_{u'' \in V(H'')} a''_{ju''} x''_{u''} + \sum_{\substack{u' \in V(H'): \\ u' \neq v'}} a'_{iu'} b''_j x'_{u'} \leq b'_i b''_j \qquad \forall i, j$$
$$x \geq 0$$

*is a complete description of the stable set polytope of $H$.*

**Theorem 4.2** (Clique identification [14])**.** *Let $H'$ and $H''$ be two graphs such that the graph $H' \cap H'' := (V(H') \cap V(H''), E(H') \cap E(H''))$ is complete and let $A'x' \leq b'$, $A''x'' \leq b''$ be complete descriptions of the stable set polytopes of $H'$ and $H''$, respectively. Then the union of these linear systems is a complete description of the stable set polytope of the graph $H' \cup H'' := (V(H') \cup V(H''), E(H') \cup E(H''))$.*

## 4.2 Connectedness of Support Dependency Graphs

The general clique polytope is *down monotone*, i.e. $x \in P$ implies $y \in P$ for all $0 \leq y \leq x$. Hence, for any nontrivial facet-defining inequality $a^\top x \leq \alpha$, we have $a \geq 0$ and can define support graphs, which are a useful tool in the study of the relations between the clique polytope and graph properties, as follows.

**Definition 4.3** (Support graph and support dependency graph)**.** *Let $G$ be a $k$-partite graph with partition $\mathcal{V} = \{V_1, \ldots, V_k\}$ of $V(G)$. Consider the clique polytope $P$ of $G$ and let $a^\top x \leq \alpha$ be a nontrivial facet-defining inequality of $P$, for which we have $a \geq 0$. We call the subgraph $G[a]$ of $G$ induced by the vertex set $\{v \in V(G) : a_v > 0\}$ the* support graph *of the inequality $a^\top x \leq \alpha$, and the dependency graph $\mathcal{G}[a]$ of the support graph $G[a]$ the* support dependency graph.

Note that the support dependency graph is not necessarily a subgraph of $\mathcal{G}$ induced by a set of vertices, as it may have fewer edges.

A clique in a graph $G$ is equivalent to a stable set in the complement graph $\overline{G}$, and the respective polytopes are in fact identical. For ease of presentation, we will later on consider

the stable set polytope on $\overline{G}$ instead of the clique polytope on $G$. Note that in this case the dependency graph is still based on $\mathcal{V}$ and $G$, but has the same meaning for stable sets in $\overline{G}$ as for cliques in $G$. Namely, for a stable set in $\overline{G}$, we cannot choose $u \in V_i$ and $v \in V_j$ independently of each other if and only if the edge $\{V_i, V_j\}$ is contained in $\mathcal{E}$. As a consequence, the support graph $\overline{G}[a]$ of a facet-defining inequality of the stable set polytope of $\overline{G}$ is the complement of the support graph $G[a]$ of the same inequality interpreted for the clique polytope on $G$, whereas the respective support dependency graphs coincide.

The following example illustrates that Theorem 4.2 cannot be extended to support dependency graphs.

**Example 4.4.** *A* wheel *of length $\ell$ is a graph that is a hole $(u_1, \dots, u_\ell)$ with an additional vertex $h$ that is connected to all vertices of the cycle. Consider the conflict graph $\overline{G}$ shown in Figure 2a, which is a wheel of length five with partition $\mathcal{V} := \{V_1, V_2, V_3, V_4\}$, where $V_1 := \{h, u_1\}, V_2 := \{u_2\}, V_3 := \{u_3, u_4\}, V_4 := \{u_5\}$. The corresponding dependency graph $\mathcal{G}$ is shown in Figure 2b. Assume $\overline{G}$ and $\mathcal{G}$ are the result of identifying the clique $\{V_1, V_3\}$ of the dependency graphs corresponding to the subgraphs $\overline{G}_1$ and $\overline{G}_2$ of $\overline{G}$ induced by $V(\overline{G}) \setminus \{u_2\}$ and $V(\overline{G}) \setminus \{u_5\}$, respectively. For $\overline{G}$, the complete description of the stable set polytope is given by*

$$\begin{aligned} 2x_h + \sum_{n=1}^{5} x_{u_n} &\leq 2 \\ \sum_{v \in C} x_v &\leq 1 \quad \forall \text{ cliques } C \text{ in } \overline{G} \\ x &\geq 0. \end{aligned} \tag{6}$$

*Note that inequality (6) is facet-defining and has nonzero coefficients for all vertices in $V(\overline{G})$. Hence it is not induced by valid inequalities of the stable set polytopes of the subgraphs $\overline{G}_1$ and $\overline{G}_2$. If we apply Theorem 4.2 three times to create an additional vertex in each of the subsets $V_1, V_2$ and $V_4$, we get the graph shown in Figure 2c. The complete description of the stable set polytope is of the same form as above. Further, inequality (6) is also facet-defining for the CPMC polytope, i.e. when assuming equality for the clique constraints*

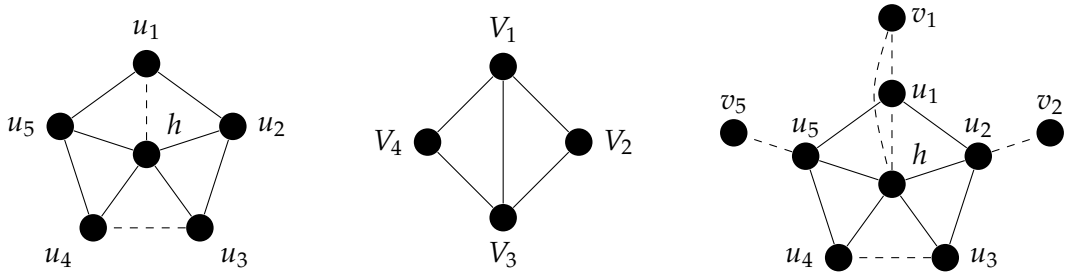
$$\begin{aligned} x_h + x_{u_1} + x_{v_1} &= 1, \\ x_{u_2} + x_{v_2} &= 1, \\ x_{u_3} + x_{u_4} &= 1, \\ x_{u_5} + x_{v_5} &= 1, \end{aligned}$$

*as it is the only inequality to cut off the point $\tilde{x}$ given by*

$$\tilde{x}_v := \begin{cases} 0, & \text{if } v = h \\ \frac{1}{2}, & \text{otherwise.} \end{cases}$$

Example 4.4 has shown that Theorem 4.2 cannot be extended to dependency graphs. However, the theorem implies that the support graph of any facet-defining inequality of the stable set polytope must be 2-connected, as any cut vertex could be seen as a 1-clique that has been used to combine the complete polyhedral descriptions of two smaller graphs into a complete polyhedral description of the whole graph. Next, we show that this is also true for support dependency graphs.

**Theorem 4.5.** *Let $(G, \mathcal{V})$ be an instance of CPMC, and let $a^\top x \leq \alpha$ be a facet-defining inequality of the stable set polytope of $\overline{G}$. Then $\mathcal{G}[a]$ is 2-connected.*




(a) Example conflict graph $\overline{G}$ (b) Dependency graph $\mathcal{G}$ (c) Conflict graph $\overline{G}'$ with additional vertices $v_1, v_2$ and $v_5$

Figure 2: Illustrations for Example 4.4. Dashed lines represent edges within subsets $V_i \in \mathcal{V}$.

*Proof.* Since $\overline{G}[a]$ is (2-)connected, its support dependency graph $\mathcal{G}[a]$ is also connected. Suppose $\mathcal{G}[a]$ is not 2-connected. Then there exists a cut vertex $V_i \in V(\mathcal{G}[a])$ such that $\mathcal{G}[a]$ decomposes into components $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ with $\mathcal{V}_1 \cap \mathcal{V}_2 = \{V_i\}$. Let $\overline{G}_1$ and $\overline{G}_2$ be the subgraphs of $\overline{G}[a]$ corresponding to $\mathcal{G}_1$ and $\mathcal{G}_2$. Then $V_i$ is a clique in $\overline{G}_1$ as well as $\overline{G}_2$, and it follows from Theorem 4.2 that the inequality $a^\top x \leq \alpha$ is induced by valid inequalities on $\overline{G}_1$ and $\overline{G}_2$. This is a contradiction and, hence, $\mathcal{G}[a]$ must be 2-connected. □

The above theorem implies that when trying to separate or even enumerate inequalities for the stable set (or clique) polytope, it suffices to consider the conflict (or compatibility) graphs represented by 2-connected subgraphs of $\mathcal{G}$. In the case where the dependency graph is a forest, this yields an alternate proof that all nontrivial facet-defining inequalities of the clique polytope of the compatibility graph $G$ can be found by looking only at the bipartite subgraphs $G_{ij}$. Even further, as bipartite graphs are themselves perfect, this also yields an alternate proof that in this case the entire compatibility graph $G$ is perfect.

In the context of CPMC, the following example shows that there is no canonical way to extend Theorem 4.5 to the CPMC polytope. However, this is not a real issue since the CPMC polytope is a face of the standard stable set polytope. Therefore, we can find the complete description of the CPMC polytope by looking at the standard stable set polytope of a given conflict graph.

**Example 4.6.** *Consider the conflict graph $\overline{G}$ shown in Figure 3a with partition $\mathcal{V} := \{V_1, V_2, V_3, V_4\}$, where $V_1 := \{u_1, v_1\}, V_2 := \{u_2, u_3\}, V_3 := \{u_4, v_4\}, := \{u_5, v_5\}, V_4 := \{u_6, u_7\}$. A complete description of the corresponding stable set polytope is given by*

$$\begin{aligned} \sum_{i=1}^{7} x_{u_i} &\leq 3 \qquad (7) \\ x_u + x_v &\leq 1 \quad \forall \{u, v\} \in E(\overline{G}) \\ x &\geq 0. \end{aligned}$$

*Similar to Example 4.4, we can consider the CPMC polytope by assuming equality for the constraints*

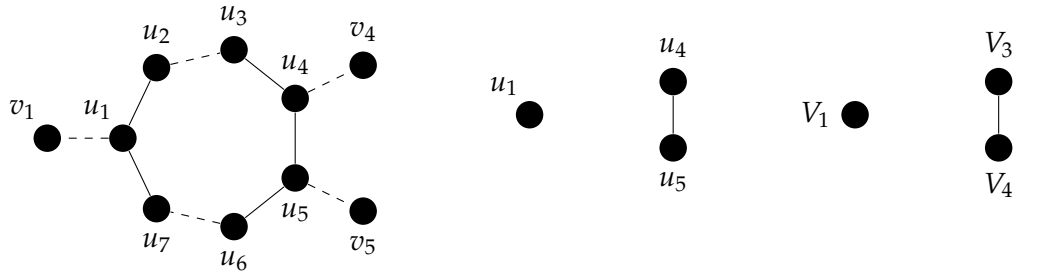
$$x_u + x_v = 1 \quad \forall \{u, v\} \in \mathcal{V},$$

*and inequality* (7) *remains facet-defining as it is the only inequality to cut off the point $\tilde{x}$ given by*

$$\tilde{x}_v := \tfrac{1}{2} \quad \forall v \in V(\overline{G}).$$

*However, the additional equality constraints allow us to simplify inequality* (7) *to*

$$x_{u_1} + x_{u_4} + x_{u_5} \leq 1.$$




(a) Conflict graph $\overline{G}$

(b) Support graph of $x_{u_1} + x_{u_4} + x_{u_5} \leq 1$

(c) Support dependency graph of $x_{u_1} + x_{u_4} + x_{u_5} \leq 1$

Figure 3: Illustrations for Example 4.6. Dashed lines represent edges within subsets $V_i \in \mathcal{V}$.

*This representation has non-negative coefficients with minimal right-hand side and is also orthogonal to the equality constraints. However, its support graph shown in Figure 3b induces the support dependency graph shown in Figure 3c, which is not 2-connected.*

# 5 Hole Dependency Graphs

We know from Theorem 4.5 that it suffices to study (sub-)graphs corresponding to 2-connected dependency graphs. Before discussing hole dependency graphs as the most basic class of 2-connected graphs, we want to present two simple reductions that can be applied to CPMC instances before solving the problem.

**Lemma 5.1** (Simple reductions)**.** *Let $(G, \mathcal{V})$ be an instance of CPMC. The following reductions may be applied:*

1. *If some subgraph $G_{ij}$ of $G$ contains an isolated vertex $v$, then we can remove $v$ from the graph $G$, and the resulting instance is feasible if and only if the original one was feasible. In the case where removing $v$ turns $G_{ij}$ into a complete bipartite graph, the edge $\{V_i, V_j\}$ is by definition no longer contained in $E(\mathcal{G})$. If a subset $V_i$ becomes empty after removing $v$, then the instance is infeasible.*

2. *If for some subgraph $G_{ij}$ of $G$ we have $|N_{G_{ij}}(v)| = 1$ for all vertices $v \in V(G_{ij})$, then we can identify $V_i$ and $V_j$, i.e.*

   *(a) for $\{u, v\} \in E(G_{ij})$ with $u \in V_i$ remove all $\{u, w\}$ from $E(G)$ where $w \notin N_G(v)$,*

   *(b) remove all vertices $v \in V_j$ from $V(G)$,*

   *(c) for $V_l \in N_{\mathcal{G}}(V_j)$ add the edges $\{V_i, V_l\}$ to $E(\mathcal{G})$,*

   *(d) remove $V_j$ from $\mathcal{V} = V(\mathcal{G})$,*

   *and the resulting instance is feasible if and only if the original one was feasible.*

*Proof.* Let $G_{ij}$ be a subgraph of $G$. A vertex in $V(G_{ij})$ with degree zero cannot be part of a feasible solution to CPMC and may therefore be removed from $G$ without changing the set of feasible solutions.

If we have $|N_{G_{ij}}(v)| = 1$ for all $v \in V(G_{ij})$, then each vertex in $V_i$ is connected to exactly one vertex in $V_j$, as $V_i$ and $V_j$ are stable sets in $G$. Hence a vertex $u \in V_i$ is part of a feasible solution to CPMC if and only if its only neighbour $v \in N_{G_{ij}}(u)$ is also part of the solution. The adjustment (a) reflects the fact that a vertex $u \in V_i$ now also represents its neighbour in $V_j$ and (b)–(d) simply ensure consistency of the instance. Therefore, the reduced instance is feasible if and only if the original instance was feasible. □

Next, we use the above lemma to identify a class of conflict graphs that turn out to be $h$-perfect. Let $(G, \mathcal{V})$ be an instance of CPMC such that for all $V_i \in \mathcal{V}$ we have $|V_i| = 2$ and the dependency graph $\mathcal{G}$ is a hole. If any vertex can be removed via the first reduction stated in Lemma 5.1, then either the instance is shown to be unsolvable or the dependency graph loses an edge, in which case we can apply the results for forest dependency graphs from [10].

**Theorem 5.2.** *Let $(G, \mathcal{V})$ be an instance of CPMC such that for all $V_i \in \mathcal{V}$ we have $|V_i| = 2$, its dependency graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a hole and none of the reductions stated in Lemma 5.1 can be applied. Then $\overline{G}$ is $h$-perfect.*

*Proof.* Consider some subgraph $\overline{G}_{ij}$ of $\overline{G}$. If the reductions from Lemma 5.1 cannot be applied to $\overline{G}_{ij}$, then we have

$$\forall v \in V(\overline{G}_{ij}) \colon |N_{\overline{G}_{ij}}(v)| \leq 2, \text{ and}$$
$$\exists v \in V(\overline{G}_{ij}) \colon |N_{\overline{G}_{ij}}(v)| = 1.$$

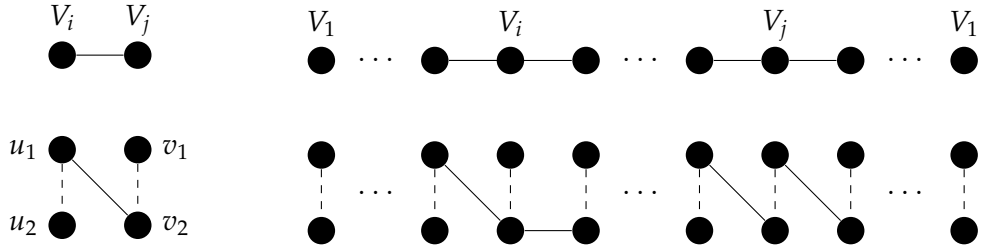
Hence all subgraphs $\overline{G}_{ij}$ of $\overline{G}$ with $\{V_i, V_j\} \in \mathcal{E}$ must be of the form shown in Figure 4a, i.e.

$$\exists! \{u, v\} \in V(\overline{G}_{ij}) \colon u \in V_i, v \in V_j.$$

Therefore, as Figure 4b illustrates with the subsets corresponding to $V_i$ and $V_j$, the graph $\overline{G}$ is exactly a hole $H$ with zero or more additional vertices of degree one. More precisely, we get a vertex of degree one for each pair of edges $\{u_1, u_2\}, \{u_2, u_3\} \in E(\overline{G})$ where $u_1, u_2$ and $u_3$ lie in three different subsets $V_i \in \mathcal{V}$. It follows from the fact that holes are $h$-perfect ([27]) in combination with Theorem 4.2 (to add the vertices of degree one) that $\overline{G}$ is also $h$-perfect, i.e. the complete description of the stable set polytope of $\overline{G}$ is given by

$$\begin{aligned} \sum_{v \in H} x_v &\leq \left\lfloor \frac{|H|}{2} \right\rfloor \\ x_u + x_v &\leq 1 \qquad \forall \{u, v\} \in E(\overline{G}) \\ x &\geq 0. \end{aligned}$$

□




(a) Shape of subgraphs $\overline{G}_{ij}$

(b) Example shape of $\overline{G}$ with vertex of degree one in $V_i$

Figure 4: Illustrations for the proof of Theorem 5.2. Dashed lines represent edges within subsets $V_i \in \mathcal{V}$.

To give a complete picture, we want to note that Theorem 5.2 can be generalized in two simple ways. First, the proof of Theorem 5.2 remains intact if the subsets in the partition $\mathcal{V}$ contain more than two elements, but for all subgraphs $\overline{G}_{ij}$ of $\overline{G}$ we have $|E(\overline{G}_{ij})| = 1$. In this case, the vertices of degree one are larger cliques, which can still be added to the original hole via Theorem 4.2.

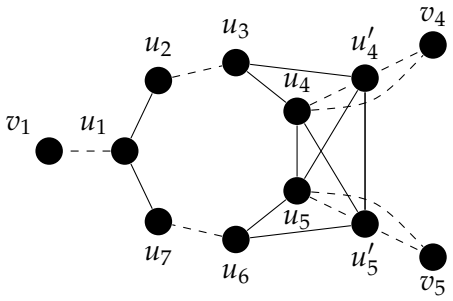



Figure 5: Conflict graph $\overline{G}$ in Example 5.3. Dashed lines represent edges within subsets $V_i \in \mathcal{V}$.

Second, we can easily generalize the result onto conflict graphs $\overline{G}$ with partition $\mathcal{V}$ such that the dependency graph is a cactus graph, i.e. every edge of $\mathcal{E}$ is part of at most one cycle. Consider two conflict graphs $\overline{G}_1$ and $\overline{G}_2$ with partitions $\mathcal{V}_1$ and $\mathcal{V}_2$ such that the respective dependency graphs are holes and there exists some $V_i \in \mathcal{V}_1 \cap \mathcal{V}_2$. Then we can apply Theorem 4.2 to identify the clique $V_i$ in $\overline{G}_1$ and $\overline{G}_2$, which does not introduce any new inequalities to the complete description of the stable set polytope of the resulting graph. On the level of dependency graphs, we have connected two cycles in a single vertex. With this operation, however, we can already construct the entire class of cactus graphs.

**Example 5.3.** *Consider the conflict graph $\overline{G}$ shown in Figure 5, which is the result of applying Theorem 4.1 to the conflict graph shown in Figure 3a, with the vertices $u_4$, $u_5$ being replaced by the 2-cliques $\{u_4, u'_4\}$ and $\{u_5, u'_5\}$ respectively. Note that the reductions stated in Lemma 5.1 cannot by applied to $\overline{G}$.*

*Consequently, the complete description of its stable set polytope is given by*

$$\begin{aligned} x_{u'_4} + x_{u'_5} + \sum_{i=1}^{7} x_{u_i} &\leq 3 \\ \sum_{v \in C} x_v &\leq 1 \quad \forall \text{ cliques } C \text{ in } \overline{G} \\ x &\geq 0. \end{aligned}$$

*Clearly, the first constraint is not a regular odd-cycle constraint, and therefore the conflict graph $\overline{G}$ is not h-perfect. Note that in this example, too, the first constraint is also facet-defining for the CPMC polytope.*

The generalized odd-cycle constraint in the previous example, henceforth referred to as *odd-clique-cycle constraint*, will be the main focus of the next section.

# 6 Odd-Clique-Cycle Constraints

In this section, we present a generalization of the well-known odd-cycle constraints, which have been studied by Padberg in [27]. Further, we state a separation algorithm for a special type of our generalization, named *embedded clique-cycle constraints*, for the case where the dependency graph is series-parallel. The runtime of our separation algorithm is polynomial for a constant number of vertices per subset $V_i \in \mathcal{V}$.

**Definition 6.1** (Clique paths)**.** *Let $G$ be a graph, and let $\mathcal{C} = \{C_1, \ldots, C_\ell, C_{\ell+1}\}$ be a set of cliques in $G$. We call $\mathcal{C}$ a* clique path *of length $\ell$ in $G$ if*

1. *the sets in $\mathcal{C}$ are pairwise disjoint and*
2. *for all $i \in \{1, \ldots, \ell\}$ we have that $C_i \cup C_{i+1}$ is a clique in $G$.*

*Further, we call $C_1$ and $C_{\ell+1}$ the* terminal cliques *of the clique path $\mathcal{C}$.*

If we interpret the vertices of a regular path as 1-cliques, then clique paths as defined above can be seen as one possible generalization of paths. Analogously, we can define the generalizations of cycles, holes and their induced constraints.

**Definition 6.2** (Clique cycles, clique holes and odd-clique-cycle constraints)**.** *Let $G$ be a graph, and let $\mathcal{C} = \{C_1, \dots, C_\ell\}$ be a clique path in $G$. We call $\mathcal{C}$ a* clique cycle *of length $\ell$ in $G$ if $C_1 \cup C_\ell$ is also a clique in $G$. For ease of notation, we consider indices modulo the cycle length $\ell$; e.g. $C_{\ell+1} = C_1$. Let $H$ be the subgraph of $G$ induced by the vertex set $\bigcup_{C\in\mathcal{C}} C$. If we have $E(H) = \bigcup_{n=1}^{\ell}\{\{u, v\} \subseteq C_n \cup C_{n+1}\}$, then we call $\mathcal{C}$ a* clique hole *in $G$.*

*Analogously to odd-cycle constraints, we can, for a given clique cycle $\mathcal{C} = \{C_1, \dots, C_\ell\}$ of odd length $\ell$, define* odd-clique-cycle (OCC) constraints *as*

$$\sum_{v\in\bigcup_{C\in\mathcal{C}} C} x_v \leq \frac{\ell-1}{2}. \tag{8}$$

**Theorem 6.3.** *Let $G$ be a graph, and let $\mathcal{C} = \{C_1, \dots, C_\ell\}$ be a clique cycle of odd length $\ell$ in $G$. Then the OCC constraint (8) is valid for the stable set polytope on $G$.*

*Proof.* By definition of $\mathcal{C}$, we have for $n \in \{1, \dots, \ell\}$ the clique constraints

$$\sum_{v\in C_n\cup C_{n+1}} x_v \leq 1,$$

and the sum of these constraints is

$$\sum_{v\in\bigcup_{C\in\mathcal{C}} C} 2x_v \leq \ell.$$

The left-hand side coefficients remain integral after dividing the inequality by two and the variables $x_v$ must be in $\{0, 1\}$ if $x$ is an incidence vector of a stable set in $G$. Hence, the left-hand side is always integral and the inequality remains valid if we round down the right-hand side. This yields exactly the OCC constraint. □

Note that for clique cycles of length three, an OCC constraint is equivalent to a regular clique constraint.

Padberg has shown in [27] that an odd-cycle constraint is facet-defining if and only if the cycle is a hole. In the case of OCC constraints, we can split this result into a sufficient and a necessary condition for the OCC constraint being facet-defining.

**Theorem 6.4.** *Let $G$ be a graph, and let $\mathcal{C} = \{C_1, \dots, C_\ell\}$ be an odd clique cycle in $G$. If there exists some odd $m \in \{3, \dots, \ell-2\}$ such that $C_1 \cup C_m$ is a clique in $G$, then the corresponding OCC constraint (8) is implied by the OCC constraint*

$$\sum_{v\in\bigcup_{C\in\mathcal{C}'} C} x_v \leq \frac{m}{2}$$

*corresponding to $\mathcal{C}' \coloneqq \{C_1, \dots, C_m\}$ together with the clique constraints*

$$\sum_{v\in C_n\cup C_{n+1}} x_v \leq 1 \quad \forall n \in \{m+1, m+3, \dots, \ell-1\}.$$

*Proof.* This follows directly from the fact that the OCC constraint corresponding to $\mathcal{C}$ is the sum of the other constraints, which are all valid, since $\mathcal{C}'$ is an odd clique cycle in $G$ and $C_n \cup C_{n+1}, n \in \{m+1, m+3, \dots, \ell-1\}$ are cliques in $G$. □

An inequality which is induced by other valid inequalities cannot be facet-defining. Hence Theorem 6.4 suggests the following definition as a necessary condition for an OCC constraint to be facet-defining.

**Definition 6.5.** *Let $G$ be a graph. We call an OCC constraint $a^\top x \leq \alpha$* irreducible *if Theorem 6.4 cannot be applied.*

Next, we will give a sufficient condition for an OCC constraint to be facet-defining by generalizing the fact that an odd-cycle constraint is facet-defining for the stable set polytope if and only if it is a hole to OCC constraints.

**Theorem 6.6.** *Let $G$ be a graph, and let $\mathcal{C} = \{C_1, \dots, C_\ell\}$ be a clique hole of odd length $\ell$ in $G$ with $\bigcup_{C \in \mathcal{C}} C = V(G)$. Then the corresponding OCC constraint is facet-defining for the stable set polytope of $G$.*

*Proof.* If we have $|C| = 1$ for all $C \in \mathcal{C}$, then the OCC constraint is a regular odd-hole constraint and therefore facet-defining. Otherwise, after possibly renaming the cliques in $\mathcal{C}$, we have $|C_1| \geq 2$. Further, let $v \in C_1$, let $G'$ be the subgraph of $G$ where all vertices in $C_1 \setminus \{v\}$ have been removed, and let $G''$ be a complete graph with $V(G'') = C_1$. Then the graph $G$ is exactly the result of substituting the remaining vertex $v$ in $G'$ by the graph $G''$. This reduction procedure can iteratively be repeated until after at most $\ell$ steps we have $|C| = 1$ for all $C \in \mathcal{C}$. In this case, we have reduced $G$ to an odd hole $H_0$ of length $\ell$, for which the complete description of the stable set polytope is given by

$$\begin{aligned} \sum_{v \in V(H_0)} x_v &\leq \frac{\ell - 1}{2} \\ \sum_{v \in C} x_v &\leq 1 \qquad \forall \text{ cliques C in } H_0 \\ x &\geq 0. \end{aligned}$$

For a complete graph $K_n$ consisting of $n$ vertices, the complete polyhedral description is given by

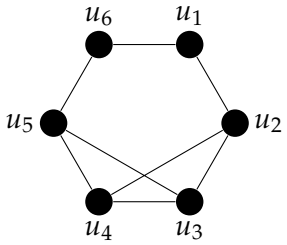
$$\begin{aligned} \sum_{v \in V(K_n)} x_v &\leq 1 \\ x &\geq 0. \end{aligned}$$

Starting with the hole $H_0$, we can reverse the reductions one by one by applying Theorem 4.1 with our current graph $H_i$ as $H'$ and a complete graph $K_{n_i}$ of fitting size as $H''$. After the $i$-th step the complete description for the new graph $H_i$ is given by

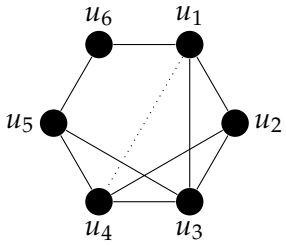
$$\begin{aligned} \sum_{v \in V(H_i)} x_v &\leq \frac{\ell - 1}{2} \qquad (9) \\ \sum_{v \in C} x_v &\leq 1 \qquad \forall \text{ cliques C in } H_i \\ x &\geq 0, \end{aligned}$$

where (9) is a facet-defining OCC constraint. In particular, this holds for the final step with $H_i = G$. □

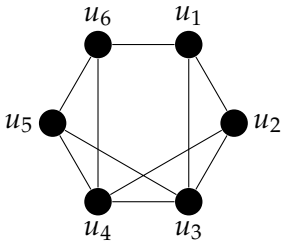
Before moving on to an example illustrating Theorems 6.4 and 6.6, we want to mention that the procedure of substituting a vertex by a clique can be applied to all known classes of facet-defining inequalities. Depending on the resulting structures of the conflict and dependency graphs, separation algorithms may also be developed for these generalizations of other constraints.




(a) The graph $G_1$ is a clique hole.




(b) Graph $G_2$ is not a clique hole, but still induces a facet-defining irreducible OCC constraint. If the dotted edge is also added, then Theorem 6.4 can be applied with $m = 3$.




(c) The same OCC constraint is irreducible in graph $G_3$, but no longer induces a facet.

Figure 6: Example graphs illustrating Theorems 6.4 and 6.6.

**Example 6.7.** *The graph $G_1$ shown in Figure 6a induces the OCC constraint*

$$\sum_{i=1}^{6} x_{u_i} \leq 2,$$

*via the clique cycle $\mathcal{C} = \{\{u_1\}, \{u_2\}, \{u_3, u_4\}, \{u_5\}, \{u_6\}\}$. In fact, $\mathcal{C}$ is a clique hole, and therefore its OCC constraint is facet-defining for the stable set polytope of $G_1$.*

*With the additional edge $\{u_1, u_3\}$, we get the graph $G_2$ shown in Figure 6b. The clique cycle $\mathcal{C}$ is no longer a clique hole, but we will see that its OCC constraint remains facet-defining. Consider the following linear system consisting of six incidence vectors of 2-cliques in $G_2$.*

$$\begin{array}{l} u_1\colon \\ u_2\colon \\ u_3\colon \\ u_4\colon \\ u_5\colon \\ u_6\colon \end{array} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

*The third row (corresponding to $u_3$) directly implies $\lambda_5 = 0$. Together with the remaining rows, we get*

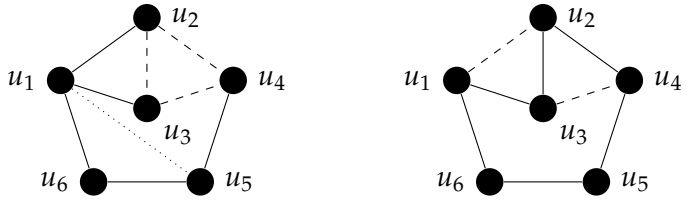
$$\lambda_1 \overset{u_1}{=} -\lambda_2 \overset{u_5}{=} \lambda_3 \overset{u_2}{=} -\lambda_4 \overset{\lambda_5=0}{\underset{u_6}{=}} \lambda_6 \overset{u_4}{=} -\lambda_1.$$

*Hence the six incidence vectors are linearly independent, which proves that the face induced by the OCC constraint is 5-dimensional and therefore a facet of the stable set polytope of $G_2$. If the edge $\{u_1, u_4\}$ is added, then Theorem 6.4 applies with $m = 3$. In this case, the OCC constraint is no longer facet-defining, as it is exactly the sum of $\sum_{i=1}^{4} x_{u_i} \leq 1$ and $x_{u_5} + x_{u_6} \leq 1$.*

*The two examples $G_1$ and $G_2$ seem to suggest that irreducibility is a sufficient condition for an OCC constraint to be facet-defining. However, the graph $G_3$ shown in Figure 6c is a counterexample. In this graph, we have the same irreducible OCC constraint, but it is no longer facet-defining, as it can be represented as the sum of the two clique constraints $\sum_{i=1}^{3} x_{u_i} \leq 1$ and $\sum_{i=4}^{6} x_{u_i} \leq 1$.*

## 6.1 Embedded OCC Constraints

Next, we consider basic properties of the class of embedded OCC constraints.




(a) Conflict graph $\overline{G}$ with partition $\mathcal{V}_1$

(b) Conflict graph $\overline{G}$ with partition $\mathcal{V}_2$

Figure 7: Illustrations for Example 6.9. Dashed lines represent edges within subsets $V_i \in \mathcal{V}$.

**Definition 6.8** (Embedded clique paths, clique cycles and OCC constraints)**.** *Let $(G, \mathcal{V})$ be an instance of CPMC, and let $\mathcal{C} = \{C_1, \dots, C_\ell\}$ be a clique path or clique cycle in $\overline{G}$. We call $\mathcal{C}$* embedded *in $\mathcal{V}$ if*

$$\forall C \in \mathcal{C} \colon \exists V_i \in \mathcal{V} \colon C \subseteq V_i.$$

*We call an OCC constraint* embedded *if its corresponding clique cycle is embedded in $\mathcal{V}$.*

For embedded OCC constraints, we can make the following two observations. Let $(G, \mathcal{V})$ be an instance of CPMC, and let $\mathcal{C} = \{C_1, \dots, C_\ell\}$ be an embedded clique cycle in $\overline{G}$. First, let $C_n, C_m \in \mathcal{C}$ with $C_n \notin \{C_{m-1}, C_m, C_{m+1}\}$. If the condition

$$\exists V_i \in \mathcal{V} \colon C_n \cup C_m \subseteq V_i$$

holds, then the OCC constraint corresponding to $\mathcal{C}$ can be reduced by Theorem 6.4. This follows directly from the fact that $V_i$ is a clique in $\overline{G}$. Second, the first observation directly implies that the support dependency graph of an irreducible embedded OCC constraint must be a cycle. Further, the support dependency graph is a hole if and only if $\mathcal{C}$ is a clique hole.

The following example illustrates Definition 6.8 and the two observations. In the next section, we provide a separation algorithm for embedded OCC constraints.

**Example 6.9.** *Consider the conflict graph $\overline{G}$ shown in Figure 7 with partitions*

$$\mathcal{V}_1 \coloneqq \{\{u_1\}, \{u_2, u_3, u_4\}, \{u_5\}, \{u_6\}\}$$
$$\mathcal{V}_2 \coloneqq \{\{u_1, u_2\}, \{u_3, u_4\}, \{u_5\}, \{u_6\}\}.$$

*The clique cycle $\mathcal{C} \coloneqq \{\{u_1\}, \{u_2, u_3\}, \{u_4\}, \{u_5\}, \{u_6\}\}$ is embedded in $\mathcal{V}_1$, but not in $\mathcal{V}_2$. In both cases, the clique cycle $\mathcal{C}$ is in fact a clique hole, and therefore*

$$\sum_{n=1}^{6} x_n \leq 2$$

*is facet-inducing for the stable set polytope of $\overline{G}$. If, however, we were to merge the subsets $\{u_1\}$ and $\{u_5\}$ in the partition $\mathcal{V}_1$, which would require adding the edge $\{u_1, u_5\}$, then we could indeed apply Theorem 6.4 to get the constraints*

$$x_{u_1} + x_{u_5} + x_{u_6} \leq 1$$
$$x_{u_2} + x_{u_3} + x_{u_4} \leq 1,$$

*implying the OCC constraint above.*

### 6.2 Separation of Embedded OCC Constraints

We have established that the support dependency graph of an embedded OCC constraint is always a cycle and the cliques are arranged in a way such that the clique cycle moves around the cycle in the dependency graph. If the dependency graph is series-parallel, then we can use these properties and state a separation algorithm for embedded OCC constraints, despite the fact that the number of cycles in a series-parallel graph may be exponential (see [16]). The basic idea is to consider candidates for clique paths an OCC constraint could consist of and to use a decomposition tree to check all required combinations of clique paths in an efficient manner. The result is a dynamic program where we iteratively process and remove a leaf of the decomposition tree until either a violated embedded OCC constraint has been found or the decomposition tree becomes empty, in which case no violated embedded OCC constraint exists. The runtime of the resulting separation algorithm is polynomial in the size of the decomposition tree as well as the number of maximal cliques in the subgraphs $\overline{G}_{ij}$. Although the latter may be exponential in the number of vertices in the conflict graph, there exist non-trivial bounds on this number, which may make enumeration tractable even in relevant real-world applications. For full details on the bounds and a real-world application where the bounds have been used successfully, we refer the reader to Sections 4 and 5 in [10].

Before we begin with the presentation of the separation algorithm itself, we want to derive one more simplification. Let $(G, \mathcal{V})$ be an instance of CPMC and assume there exists an embedded OCC constraint to which Theorem 6.4 can be applied. If this constraint is violated by some $\tilde{x} \in \mathbb{R}^{|V(G)|}$, then at least one of the following two statements holds:

1. There exists, for fitting subsets $V_i, V_j \in \mathcal{V}$, some clique $C \subseteq V_i \cup V_j$ with $\sum_{v \in C} \tilde{x}_v > 1$.
2. There exists an irreducible OCC constraint that is also violated.

The first statement can be checked by separating the stable set constraints over all bipartite subgraphs $G_{ij}$ of the compatibility graph $G$. As shown in the last paragraph in Section 4 in [10], this can be done in $\mathcal{O}(|\mathcal{V}| \cdot |V_{\max}|^3)$ time, where $V_{\max} \in \mathcal{V}$ is a subset of maximum cardinality. In general, this will be faster than our separation algorithm, so we assume w.l.o.g. that the first statement is false for the vector $\tilde{x} \in \mathbb{R}^{|V(G)|}$. With this assumption, however, we can restrict ourselves to separating irreducible embedded OCC constraints, as this is now equivalent to separation of all embedded OCC constraints.

An important building block of our separation algorithm will be the slack of the following constraint, corresponding to a given clique path $C_1, \ldots, C_{\ell+1}$:

$$\sum_{v \in \bigcup_{n=2}^{\ell} C_n} x_v \leq \left\lfloor \frac{\ell}{2} \right\rfloor. \tag{10}$$

Constraint (10) is clearly valid for the stable set polytope, as it is simply the sum of the clique constraints corresponding to the cliques $C_n \cup C_{n+1}, n \in \{1, \ldots, \ell\}$, where after division by two, the coefficients of all $x_v$ with $v \in C_1 \cup C_{\ell+1}$ have been set to zero.

Let $(G, \mathcal{V})$ be an instance of CPMC with series-parallel dependency graph $\mathcal{G}$ and $\mathcal{T}$ a decomposition tree of $\mathcal{G}$ as introduced in Definition 2.2. Further, let $\tilde{x} \in \mathbb{R}^{|V(G)|}$ be some vector such that for all subgraphs $\overline{G}_{ij}$ of $\overline{G}$ the inequalities

$$\sum_{v \in C} \tilde{x}_v \leq 1 \quad \forall \text{ cliques } C \text{ in } \overline{G}_{ij} \tag{11}$$

are satisfied. The task is then to decide if $\tilde{x}$ satisfies all embedded OCC constraints, which is now equivalent to satisfying all irreducible embedded OCC constraints.

Consider some node $e \in V(\mathcal{T})$ with terminals $V_s$ and $V_t$ and let $\overline{G}_e, \mathcal{G}_e$ be the corresponding conflict and dependency graph, respectively, represented by the node $e$. Further, we define the

sets of clique terminals by

$$S_e \coloneqq \{C \cap V_s : C \text{ is a maximal clique in } \overline{G}_e \text{ with } |\{V_i \in V(\mathcal{G}_e) \colon C \cap V_i \neq \emptyset\}| = 2\} \setminus \{\emptyset\},$$
$$T_e \coloneqq \{C \cap V_t : C \text{ is a maximal clique in } \overline{G}_e \text{ with } |\{V_i \in V(\mathcal{G}_e) \colon C \cap V_i \neq \emptyset\}| = 2\} \setminus \{\emptyset\}.$$

For $(C_s, C_t, p) \in S_e \times T_e \times \{0,1\}$, we define $\mathcal{P}_e(C_s, C_t, p)$ as the set of embedded clique paths $\mathcal{C} = \{C_1, \dots, C_{\ell+1}\}$ in $\overline{G}_e$ with terminal cliques $C_s = C_1$ and $C_t = C_{\ell+1}$ satisfying the following conditions:

$$\forall C_n \in \{C_2, \dots, C_\ell\} \colon C_n \cap (V_s \cup V_t) = \emptyset \tag{12}$$
$$\ell \equiv p \pmod 2 \tag{13}$$
$$\forall V_i \in \mathcal{V} \colon |\{C_n \in \mathcal{C} \colon C_n \cap V_i \neq \emptyset\}| \leq 2. \tag{14}$$

By definition, the clique paths $\mathcal{C} \in \mathcal{P}_e(C_s, C_t, p)$ start and end in $C_s \subseteq V_s$ and $C_t \subseteq V_t$, respectively, and do not move within the terminals $V_s, V_t$ due to condition (12). Further, the clique paths are of length parity $p$ by condition (13). Finally, all clique paths excluded by condition (14) can be reduced via Theorem 6.4. The sets $\mathcal{P}_e(C_s, C_t, p)$ then contain all relevant clique paths in the sense that all irreducible embedded OCC constraints are the composition of two clique paths $\mathcal{C}_1 \in \mathcal{P}_{e_1}(C_{s_1}, C_{t_1}, p_1)$ and $\mathcal{C}_2 \in \mathcal{P}_{e_2}(C_{s_2}, C_{t_2}, p_2)$, where $e_1$ and $e_2$ are the children of some node $e \in V(\mathcal{T})$ representing a parallel composition. Example 6.10 illustrates the definitions above.

**Example 6.10.** *Consider the conflict graph shown in Figure 8 together with its series-parallel dependency graph $\mathcal{G}_e$ with terminals $V_s$ and $V_t$ represented by some decomposition tree node $e$. Then we get the clique terminal sets*

$$S_e = \{\{s_1\}, \{s_2\}\}, T_e = \{\{t_1, t_2\}, \{t_3\}\}.$$

*For the sets of connecting clique paths we then get*

$$\begin{aligned}
&\mathcal{P}_e(\{s_1\}, \{t_1, t_2\}, 0) = \{\mathcal{C}_1 \coloneqq \{\{s_1\}, \{i_1\}, \{t_1, t_2\}\}\},\\
&\mathcal{P}_e(\{s_1\}, \{t_3\}, 0) = \emptyset,\\
&\mathcal{P}_e(\{s_2\}, \{t_1, t_2\}, 0) = \emptyset,\\
&\mathcal{P}_e(\{s_2\}, \{t_3\}, 0) = \{\mathcal{C}_2 \coloneqq \{\{s_2\}, \{i_2\}, \{t_3\}\}\},\\
&\mathcal{P}_e(\{s_1\}, \{t_1, t_2\}, 1) = \emptyset,\\
&\mathcal{P}_e(\{s_1\}, \{t_3\}, 1) = \{\mathcal{C}_3 \coloneqq \{\{s_1\}, \{i_1\}, \{i_2\}, \{t_3\}\}\},\\
&\mathcal{P}_e(\{s_2\}, \{t_1, t_2\}, 1) = \{\mathcal{C}_4 \coloneqq \{\{s_2\}, \{i_2\}, \{i_1\}, \{t_1, t_2\}\}\},\\
&\mathcal{P}_e(\{s_2\}, \{t_3\}, 1) = \emptyset.
\end{aligned}$$

*Note that some of the sets are empty as there exists no clique path fulfilling the required conditions. Further, it is indeed not necessary to consider all combinations of terminal cliques, but only those contained in $S_e \times T_e$. For instance, an OCC constraint containing a clique path with terminal cliques $\{s_1\}$ and $\{t_2\}$ can be built from the clique path $\mathcal{C}_1$. Later on we will see that, should a case like this arise, the terminal clique $\{t_1, t_2\}$ of $\mathcal{C}_1$ will be adjusted accordingly.*

The core of our algorithm is to calculate for all $(C_s, C_t, p) \in S_e \times T_e \times \{0,1\}$ the minimal slack of all constraints (10) corresponding to clique paths $\{C_1, \dots, C_{\ell+1}\} \in \mathcal{P}_e(C_s, C_t, p)$, i.e. it is required to calculate

$$f_e(C_s, C_t, p) \coloneqq \min_{\{C_1, \dots, C_{\ell+1}\} \in \mathcal{P}(C_s, C_t, p)} \left\lfloor \frac{\ell}{2} \right\rfloor - \sum_{v \in \bigcup_{n=2}^{\ell} C_n} \tilde{x}_v.$$

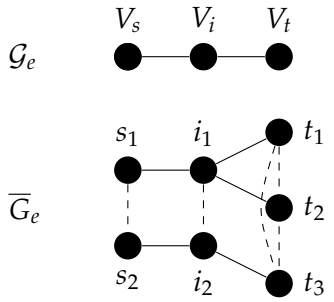



Figure 8: Illustration for Example 6.10 Dashed lines represent edges within the subsets $V_i \in \mathcal{V}$.

Note that $f_e(C_s, C_t, p)$ is always non-negative due to condition (11).

We will now describe how these values can be computed for all nodes of the decomposition tree $\mathcal{T}$. For parallel compositions, we also describe how a pair of clique paths can be combined to form a clique cycle. In each step, we consider some leaf $e \in V(\mathcal{T})$ with terminals $V_s, V_t \in \mathcal{V}$, representing either an edge $\{V_s, V_t\}$ or the composition of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ corresponding to two children $e_1, e_2 \in V(\mathcal{T})$ of $e$ with terminal clique sets $S_{e_1}$, $T_{e_1}$ and $S_{e_2}$, $T_{e_2}$, respectively. Once a leaf $e$ is processed, it is removed from the decomposition tree $\mathcal{T}$. In the following, we will elaborate on each of the four resulting cases.

**Base case** In this case, the leaf $e \in V(\mathcal{T})$ with terminals $V_s$, $V_t$ represents the graph consisting of only the edge $\{V_s, V_t\}$. Since the corresponding conflict graph is exactly $\overline{G}_{st}$, we have

$$S_e = \{C \cap V_s : C \text{ is a maximal clique in } \overline{G}_{st} \text{ with } C \cap V_s \neq \emptyset \text{ and } C \cap V_t \neq \emptyset\},$$
$$T_e = \{C \cap V_t : C \text{ is a maximal clique in } \overline{G}_{st} \text{ with } C \cap V_s \neq \emptyset \text{ and } C \cap V_t \neq \emptyset\}.$$

Let $(C_s, C_t, p) \in S_e \times T_e \times \{0, 1\}$. The set $\mathcal{P}_e(C_s, C_t, p)$ is empty if $p = 0$ or $C_s \cup C_t$ is not a clique in $\overline{G}_{st}$ and otherwise contains exactly the clique path $\{C_s, C_t\}$. Consequently, we get

$$f_e(C_s, C_t, p) = \begin{cases} 0, & \text{if } (p = 1) \wedge (C_s \cup C_t \text{ is a clique in } \overline{G}_{st}) \\ \infty, & \text{otherwise.} \end{cases}$$

**Series composition 'S'** A series composition cannot introduce any new cycles in the dependency graph and, hence, it cannot introduce any new irreducible embedded OCC constraints. Therefore, it suffices to calculate $f_e(C_s, C_t, p)$ for all $(C_s, C_t, p) \in T_e \times S_e \times \{0, 1\}$, where $S_e = S_{e_1}$ and $T_e = T_{e_2}$.

To do this, we first introduce the two functions $g_e(C_s, C', C'', C_t)$ and $h_e(C_s, C', C'', C_t)$ to calculate for a given choice of terminal cliques $(C_s, C_t) \in S_e \times T_e$ and intermediate terminal cliques $(C', C'') \in T_{e_1} \times S_{e_2}$ the minimal slack of inequality (10) of any clique path $C_1, \ldots, C_{\ell+1}$ composed of two clique paths $\mathcal{C}_1 \in \mathcal{P}_{e_1}(C_s, C', 0) \cup \mathcal{P}_{e_1}(C_s, C', 1)$ and $\mathcal{C}_2 \in \mathcal{P}_{e_2}(C'', C_t, 0) \cup \mathcal{P}_{e_2}(C'', C_t, 1)$, which has even ($g_e$) or odd ($h_e$) total length $\ell$. The composition of $\mathcal{C}_1$ and $\mathcal{C}_2$ may take place in two ways. First, if the intermediate terminal cliques $C'$ and $C''$ do not intersect, then we compose $\mathcal{C}_1$ and $\mathcal{C}_2$ by concatenation, i.e. $\mathcal{C} = \{C_s, \ldots, C', C'', \ldots, C_t\}$, and the length of the composed clique path is equal to the sum of the lengths of $\mathcal{C}_1$ and $\mathcal{C}_2$ plus one. Second, if the intermediate terminal cliques $C'$ and $C''$ do intersect, then we compose $\mathcal{C}_1$ and $\mathcal{C}_2$ by identification, i.e. $\mathcal{C} = \{C_s, \ldots, C' \cap C'', \ldots, C_t\}$, and the resulting length is equal to the sum of the lengths of $\mathcal{C}_1$ and $\mathcal{C}_2$. As a result, the slack of inequality (10) must be updated by the term $\sum_{v \in C' \cup C''} \tilde{x}_v$ in the first case, and by $\sum_{v \in C' \cap C''} \tilde{x}_v$ in the second case. Depending on the parity of the clique paths $\mathcal{C}_1$ and $\mathcal{C}_2$ as well as whether the composition is done by concatenation or

identification, we have to subtract one to update the right-hand side term $\lfloor \frac{\ell}{2} \rfloor$ in inequality (10):

$$g_e(C_s, C', C'', C_t) :=$$
$$\begin{cases} \min\{f_{e_1}(C_s, C', 0) + f_{e_2}(C'', C_t, 0), \\ \qquad f_{e_1}(C_s, C', 1) + f_{e_2}(C'', C_t, 1) + 1\} - \sum_{v \in C' \cap C''} \tilde{x}_v, & \text{if } C' \cap C'' \neq \emptyset \\ \min\{f_{e_1}(C_s, C', 0) + f_{e_2}(C'', C_t, 1), \\ \qquad f_{e_1}(C_s, C', 1) + f_{e_2}(C'', C_t, 0)\} - \sum_{v \in C' \cup C''} \tilde{x}_v + 1, & \text{otherwise,} \end{cases}$$
$$h_e(C_s, C', C'', C_t) :=$$
$$\begin{cases} \min\{f_{e_1}(C_s, C', 0) + f_{e_2}(C'', C_t, 1), \\ \qquad f_{e_1}(C_s, C', 1) + f_{e_2}(C'', C_t, 0)\} - \sum_{v \in C' \cap C''} \tilde{x}_v, & \text{if } C' \cap C'' \neq \emptyset \\ \min\{f_{e_1}(C_s, C', 0) + f_{e_2}(C'', C_t, 0), \\ \qquad f_{e_1}(C_s, C', 1) + f_{e_2}(C'', C_t, 1) + 1\} - \sum_{v \in C' \cup C''} \tilde{x}_v, & \text{otherwise.} \end{cases}$$

With the help of the auxiliary functions $g_e(C_s, C', C'', C_t)$ and $h_e(C_s, C', C'', C_t)$, which can be calculated in constant time, we can now calculate for $(C_s, C_t, p) \in S_e \times T_e \times \{0, 1\}$ the minimal slack of all constraints (10) corresponding to clique paths $\mathcal{C} \in \mathcal{P}_e(C_s, C_t, e)$ by taking the minimum over all possible combinations of intermediate terminal cliques $(C', C'') \in T_{e_1} \times S_{e_2}$ to get

$$f_e(C_s, C_t, p) := \begin{cases} \min_{C' \in T_{e_1}, C'' \in S_{e_2}} g_e(C_s, C', C'', C_t), & \text{if } p = 0 \\ \min_{C' \in T_{e_1}, C'' \in S_{e_2}} h_e(C_s, C', C'', C_t), & \text{otherwise.} \end{cases}$$

**Example 6.11** (Example series composition 'S')**.** *Consider the dependency graph $\mathcal{G}_e$ and conflict graph $\overline{G}_e$ represented by a node e of a decomposition tree $\mathcal{T}$ shown in Figure 9. Further, let e with terminals $V_s$ and $V_t$ represent the parallel composition of the graphs $\mathcal{G}_{e_1} = (\{V_s, V_i\}, \{\{V_s, V_i\}\})$ and $\mathcal{G}_{e_2} = (\{V_i, V_j, V_t\}, \{\{V_i, V_j\}, \{V_j, V_t\}\})$ with terminals $V_s, V_i$ and $V_i, V_j$, respectively. For the sets of terminal cliques, we have*

$$S_e = S_{e_1} = \{\{s_1\}, \{s_2, s_3\}\},\ T_{e_1} = \{\{i_1\}, \{i_2, i_3\}\} \text{ and}$$
$$S_{e_2} = \{\{i_2, i_3\}\},\ T_e = T_{e_2} = \{\{t_1\}\}.$$

*Further, we already have*

$$f_{e_1}(C_s, C_t, 0) = \infty \qquad \forall (C_s, C_t) \in S_{e_1} \times T_{e_1}$$
$$f_{e_1}(\{s_1\}, \{i_1\}, 1) = 0$$
$$f_{e_1}(\{s_1\}, \{i_2, i_3\}, 1) = \infty$$
$$f_{e_1}(\{s_2, s_3\}, \{i_1\}, 1) = \infty$$
$$f_{e_1}(\{s_2, s_3\}, \{i_2, i_3\}, 1) = 0$$
$$f_{e_2}(\{i_2, i_3\}, \{t_1\}, 0) = \lfloor \tfrac{2}{2} \rfloor - \tilde{x}_{j_1} = \tfrac{3}{4}$$
$$f_{e_2}(\{i_2, i_3\}, \{t_1\}, 1) = \infty.$$

*In the graph $\overline{G}_e$, we have two clique paths, namely*

$$\mathcal{C}_1 = \{\{s_1\}, \{i_1\}, \{i_2, i_3\}, \{j_1\}, \{t_1\}\} \text{ and}$$
$$\mathcal{C}_2 = \{\{s_2, s_3\}, \{i_2, i_3\}, \{j_1\}, \{t_1\}\},$$

*that are maximal in the sense that they are not component-wise contained in other clique paths contained in any $\mathcal{P}_e(C_s, C_t, p)$ with $(C_s, C_t, p) \in S_e \times T_e \times \{0, 1\}$. The first clique path $\mathcal{C}_1$ is covered by the*

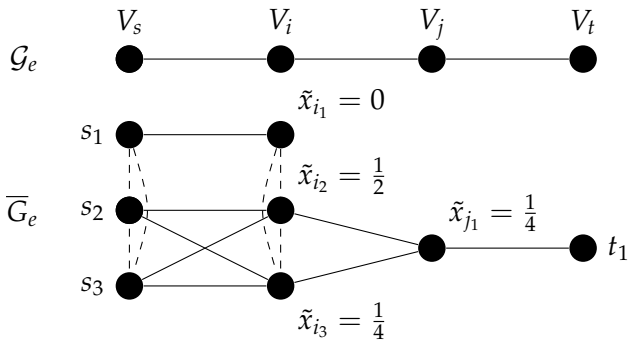



Figure 9: Graphs represented by decomposition tree node $e$ with vector $\tilde{x} \in \mathbb{R}^8$ to be separated by an OCC constraint. Dashed lines represent edges within the subsets $V_i \in \mathcal{V}$.

*combination* $(\{s_1\}, \{i_1\}, \{i_2, i_3\}, \{t_1\}) \in S_{e_1} \times T_{e_1} \times S_{e_2} \times T_{e_2}$*, for which we get*

$$\begin{aligned} g_e(\{s_1\}, \{i_1\}, \{i_2, i_3\}, \{t_1\}) &= \min\{f_{e_1}(\{s_1\}, \{i_1\}, 0) + f_{e_2}(\{i_2, i_3\}, \{t_1\}, 1), \\ &\qquad f_{e_1}(\{s_1\}, \{i_1\}, 1) + f_{e_2}(\{i_2, i_3\}, \{t_1\}, 0)\} - \bigcup_{v \in V_i} \tilde{x}_v + 1 \\ &= \min\{\infty + \infty, 0 + \tfrac{3}{4}\} - \tfrac{3}{4} + 1 \\ &= 1. \end{aligned}$$

*For all other combinations* $(C_s, C', C'', C_t)$*, we simply get* $g_e(C_s, C', C'', C_t) = \infty$*, since the required clique paths do not exist. Hence, we have*

$$f_e(\{s_1\}, \{t_1\}, 0) = 1 \quad \textit{and} \quad f_e(\{s_2, s_3\}, \{t_1\}, 0) = \infty,$$

*which is exactly the required minimum slack of inequality* (10)*.*

*The clique path* $\mathcal{C}_2$ *is covered by* $(\{s_2, s_3\}, \{i_2, i_3\}, \{i_2, i_3\}, \{t_1\}) \in S_{e_1} \times T_{e_1} \times S_{e_2} \times T_{e_2}$*, and here we get*

$$\begin{aligned} h_e(\{s_2, s_3\}, \{i_2, i_3\}, \{i_2, i_3\}, \{t_1\}) &= \min\{f_{e_1}(\{s_2, s_3\}, \{i_2, i_3\}, 0) + f_{e_2}(\{i_2, i_3\}, \{t_1\}, 1), \\ &\qquad f_{e_1}(\{s_2, s_3\}, \{i_2, i_3\}, 1) + f_{e_2}(\{i_2, i_3\}, \{t_1\}, 0)\} - \tilde{x}_{i_2} - \tilde{x}_{i_3} \\ &= \min\{\infty + \infty, 0 + \tfrac{3}{4}\} - \tfrac{3}{4} \\ &= 0. \end{aligned}$$

*Again, for all other combinations* $(C_s, C', C'', C_t)$ *we simply get* $h_e(C_s, C', C'', C_t) = \infty$*, since the required clique paths do not exist. Hence we have*

$$f_e(\{s_2, s_3\}, \{t_1\}, 1) = 0 \quad \textit{and} \quad f_e(\{s_1\}, \{t_1\}, 1) = \infty,$$

*which is, once more, exactly the required minimum slack of inequality* (10)*. Note that this indeed covers all relevant clique paths as, for example, the clique path* $\{\{s_3\}, \{i_3\}, \{j_1\}, \{t_1\}\}$ *is component-wise contained in* $\mathcal{C}_2$ *and therefore cannot induce a smaller slack. Furthermore, a clique path consisting of e.g.* $\{s_1\}, \{i_1\}, \{i_2\}, \{i_3\}, \{j_1\}$ *and* $\{t_1\}$ *is irrelevant, as it will never yield an irreducible OCC constraint due to the edge* $\{i_1, i_3\} \in E(\overline{G}_e)$*.*

**Parallel composition 'P'** This is the only composition that may introduce new cycles in the dependency graph that contain a violated (irreducible) embedded OCC constraint.

Let $\mathcal{C} = \{C_1, \ldots, C_{\ell+1}\} \in \mathcal{P}_{e_1}(C_s, C_t, 0) \cup \mathcal{P}_{e_1}(C_s, C_t, 1)$ with $(C_s, C_t) \in S_{e_1} \times T_{e_1}$, and let $\mathcal{C}' = \{C'_1, \ldots, C'_{\ell'+1}\} \in \mathcal{P}_{e_2}(C'_s, C'_t, 0) \cup \mathcal{P}_{e_2}(C'_s, C'_t, 1)$ with $(C'_s, C'_t) \in S_{e_2} \times T_{e_2}$ be clique paths such

that the slacks of their corresponding constraints (10) are minimal. More precisely, we have

$$\left\lfloor \frac{\ell}{2} \right\rfloor - \sum_{v \in \bigcup_{n=2}^{\ell} C_n} \tilde{x}_v = f_{e_1}\left(C_s, C_t, \left\lfloor \frac{\ell+1}{2} \right\rfloor - \left\lfloor \frac{\ell}{2} \right\rfloor\right) \text{ and}$$
$$\left\lfloor \frac{\ell'}{2} \right\rfloor - \sum_{v \in \bigcup_{n=2}^{\ell'} C'_n} \tilde{x}_v = f_{e_2}\left(C'_s, C'_t, \left\lfloor \frac{\ell'+1}{2} \right\rfloor - \left\lfloor \frac{\ell'}{2} \right\rfloor\right).$$

From these clique paths, we can construct a clique cycle $\mathcal{C}^\circ$ by connecting $C_s$ with $C'_s$ and $C_t$ with $C'_t$ by the same rules as in the composition of clique paths into a new clique path, i.e. by concatenation if the terminal cliques do not intersect and identification otherwise. Since clique cycles of even length are irrelevant, we assume the resulting length $|\mathcal{C}^\circ|$ of the clique cycle to be odd. Consider the four conditions

$$\begin{aligned}
&\ell \text{ is odd},\\
&\ell' \text{ is odd},\\
&C_s \cap C'_s = \emptyset,\\
&C_t \cap C'_t = \emptyset.
\end{aligned}$$

In order for $|\mathcal{C}^\circ|$ to be odd, exactly one or exactly three of these conditions must hold, and we set

$$c_{\text{length}} \coloneqq \begin{cases} 1, & \text{if three of the above conditions hold} \\ 0, & \text{otherwise} \end{cases}$$

as well as, for $(C, C') \in \{(C_s, C'_s), (C_t, C'_t)\}$, set

$$c_{C,C'} \coloneqq \begin{cases} \sum_{v \in C \cap C'} \tilde{x}_v, & \text{if } C \cap C' \neq \emptyset \\ \sum_{v \in C \cup C'} \tilde{x}_v, & \text{otherwise.} \end{cases}$$

With these corrective terms, we can calculate the slack

$$\frac{|\mathcal{C}^\circ| - 1}{2} - \sum_{v \in \bigcup_{n=1}^{|\mathcal{C}^\circ|} C_n} \tilde{x}_v$$

of the OCC constraint corresponding to $\mathcal{C}^\circ$ via

$$f_{e_1}\left(C_s, C_t, \left\lfloor \frac{\ell+1}{2} \right\rfloor - \left\lfloor \frac{\ell}{2} \right\rfloor\right) + f_{e_2}\left(C'_s, C'_t, \left\lfloor \frac{\ell'+1}{2} \right\rfloor - \left\lfloor \frac{\ell'}{2} \right\rfloor\right) + c_{\text{length}} - c_{C_s,C'_s} - c_{C_t,C'_t}.$$

For all terminal clique combinations $(C_s, C_t, C'_s, C'_t) \in S_{e_1} \times T_{e_1} \times S_{e_2} \times T_{e_2}$, we can hence determine the minimal slack of any OCC constraint composed of clique paths $\mathcal{C} \in \mathcal{P}_{e_1}(C_s, C_t, 0) \cup \mathcal{P}_{e_1}(C_s, C_t, 1)$ and $\mathcal{C}' \in \mathcal{P}_{e_2}(C'_s, C'_t, 0) \cup \mathcal{P}_{e_2}(C'_s, C'_t, 1)$ via

$$F_e(C_s, C_t, C'_s, C'_t) \coloneqq$$
$$\begin{cases}
\min\{f_{e_1}(C_s, C_t, 0) + f_{e_2}(C'_s, C'_t, 0), \\
\qquad f_{e_1}(C_s, C_t, 1) + f_{e_2}(C'_s, C'_t, 1) + 1\} - \sum_{v \in (C_s \cup C'_s) \cup (C_t \cap C'_t)} \tilde{x}_v, & \text{if } C_s \cap C'_s = \emptyset \wedge C_t \cap C'_t \neq \emptyset \\
\min\{f_{e_1}(C_s, C_t, 0) + f_{e_2}(C'_s, C'_t, 0), \\
\qquad f_{e_1}(C_s, C_t, 1) + f_{e_2}(C'_s, C'_t, 1) + 1\} - \sum_{v \in (C_s \cap C'_s) \cup (C_t \cup C'_t)} \tilde{x}_v, & \text{if } C_s \cap C'_s \neq \emptyset \wedge C_t \cap C'_t = \emptyset \\
\min\{f_{e_1}(C_s, C_t, 0) + f_{e_2}(C'_s, C'_t, 1), \\
\qquad f_{e_1}(C_s, C_t, 1) + f_{e_2}(C'_s, C'_t, 0)\} - \sum_{v \in (C_s \cap C'_s) \cup (C_t \cap C'_t)} \tilde{x}_v, & \text{if } C_s \cap C'_s \neq \emptyset \wedge C_t \cap C'_t \neq \emptyset \\
\min\{f_{e_1}(C_s, C_t, 0) + f_{e_2}(C'_s, C'_t, 1), \\
\qquad f_{e_1}(C_s, C_t, 1) + f_{e_2}(C'_s, C'_t, 0)\} - \sum_{v \in (C_s \cup C'_s) \cup (C_t \cup C'_t)} \tilde{x}_v + 1, & \text{if } C_s \cap C'_s = \emptyset \wedge C_t \cap C'_t = \emptyset.
\end{cases}$$

As a result, an irreducible embedded OCC constraint violated by $\tilde{x}$ exists in the conflict graph $\overline{G}_e$ if and only if

$$\min_{\substack{(C_s,C_t)\in S_1\times T_1\\(C'_s,C'_t)\in S_2\times T_2}} F_e(C_s,C_t,C'_s,C'_t) < 0.$$

**Example 6.12** (Example parallel composition 'P')**.** *Consider the dependency graph $\mathcal{G}_e$ and conflict graph $\overline{G}_e$ represented by a node e of a decomposition tree $\mathcal{T}$ shown in Figure 10. Further, let e with terminals $V_s$ and $V_t$ represent the parallel composition of the graphs $\mathcal{G}_{e_1} = (\{V_s, V_i, V_t\}, \{\{V_s, V_i\}, \{V_i, V_t\}\})$ and $\mathcal{G}_{e_2} = (\{V_s, V_t\}, \{\{V_s, V_t\}\})$ with the same terminals. For the sets of terminal cliques, we get*

$$\begin{aligned}&S_{e_1} = \{\{s_1, s_2\}\},\ T_{e_1} = \{\{t_1, t_2\}, \{t_3\}\} \text{ and}\\ &S_{e_2} = \{\{s_3\}\},\ T_{e_2} = \{\{t_3\}\}.\end{aligned}$$

*Further, we have*

$$\begin{aligned}&f_{e_1}(\{s_1, s_2\}, \{t_1, t_2\}, 0) = \lfloor \tfrac{2}{2} \rfloor - \tilde{x}_{i_1} = \tfrac{1}{2}\\ &f_{e_1}(\{s_1, s_2\}, \{t_3\}, 0) = \infty\\ &f_{e_1}(\{s_1, s_2\}, \{t_1, t_2\}, 1) = \infty\\ &f_{e_1}(\{s_1, s_2\}, \{t_3\}, 1) = \lfloor \tfrac{3}{2} \rfloor - \tilde{x}_{i_1} - \tilde{x}_{i_2} = 0\\ &f_{e_2}(\{s_3\}, \{t_3\}, 0) = \infty\\ &f_{e_2}(\{s_3\}, \{t_3\}, 1) = \lfloor \tfrac{1}{2} \rfloor = 0.\end{aligned}$$

*In the graph $\overline{G}_e$, we have two clique cycles, namely*

$$\begin{aligned}&\mathcal{C}_1 = \{\{s_1, s_2\}, \{i_1\}, \{t_1, t_2\}, \{t_3\}, \{s_3\}\} \text{ and}\\ &\mathcal{C}_2 = \{\{s_1, s_2\}, \{i_1\}, \{i_2\}, \{t_3\}, \{s_3\}\},\end{aligned}$$

*that are maximal in the sense that they are not component-wise contained in a different clique cycle. The first cycle $\mathcal{C}_1$ is covered by the combination $(\{s_1, s_2\}, \{t_1, t_2\}, \{s_3\}, \{t_3\}) \in S_{e_1} \times T_{e_1} \times S_{e_2} \times T_{e_2}$, for which we get*

$$\begin{aligned}F_e(\{s_1, s_2\}, \{t_1, t_2\}, \{s_3\}, \{t_3\}) &= \min\{f_{e_1}(\{s_1, s_2\}, \{t_1, t_2\}, 0) + f_{e_2}(\{s_3\}, \{t_3\}, 1),\\ &\qquad f_{e_1}(\{s_1, s_2\}, \{t_1, t_2\}, 1) + f_{e_2}(\{s_3\}, \{t_3\}, 0)\} - \bigcup_{v\in V_s\cup V_t} \tilde{x}_v + 1\\ &= \min\{\tfrac{1}{2} + 0, \infty + \infty\} - \tfrac{3}{2} + 1\\ &= 0.\end{aligned}$$

*Therefore, the OCC constraint corresponding to $\mathcal{C}_1$ is tight, but not violated. However, for the clique cycle $\mathcal{C}_2$ covered by $(\{s_1, s_2\}, \{t_3\}, \{s_3\}, \{t_3\}) \in S_{e_1} \times T_{e_1} \times S_{e_2} \times T_{e_2}$, we get*

$$\begin{aligned}F_e(\{s_1, s_2\}, \{t_3\}, \{s_3\}, \{t_3\}) &= \min\{f_{e_1}(\{s_1, s_2\}, \{t_3\}, 0) + f_{e_2}(\{s_3\}, \{t_3\}, 0),\\ &\qquad f_{e_1}(\{s_1, s_2\}, \{t_3\}, 1) + f_{e_2}(\{s_3\}, \{t_3\}, 1) + 1\} - \bigcup_{v\in V_s\cup\{t_3\}} \tilde{x}_v\\ &= \min\{\infty, 0 + 0 + 1\} - \tfrac{3}{2}\\ &= -\tfrac{1}{2}.\end{aligned}$$

*Therefore, the algorithm correctly identifies the OCC constraint of $\mathcal{C}_2$ to be violated by $\tilde{x}$.*

We want to note that our algorithm makes no distinction between OCC constraints belonging to clique cycles of length three and regular clique constraints. For instance, if the edge $\{s_3, i_2\}$

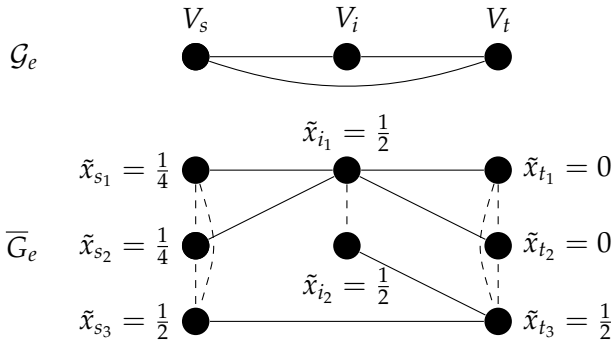



Figure 10: Graphs represented by decomposition tree node $e$ with vector $\tilde{x} \in \mathbb{R}^8$ to be separated by an OCC constraint. Dashed lines represent edges within subsets $V_i \in \mathcal{V}$.

was contained in $E(\overline{G})$ in the example above, then our algorithm would identify the (violated) constraint

$$x_{s_3} + x_{i_2} + x_{t_3} \leq 1,$$

which is exactly a clique constraint.

If the clique-path compositions described above do not yield a violated irreducible embedded OCC constraint, then we know from condition (11) and Theorem 6.4 that $\tilde{x}$ does not violate any embedded OCC constraint, irreducible or not, whose support graph is an induced subgraph of $\overline{G}_e$, and we proceed moving through the decomposition tree after the following updates.

First, the new sets of terminal cliques are given by $S_e = S_{e_1} \cup S_{e_2}$ and $T_e = T_{e_1} \cup T_{e_2}$. Second, for all $(C_s, C_t, p) \in S_e \times T_e \times \{0, 1\}$, we get

$$f_e(C_s, C_t, p) = \begin{cases} \max\{f_{e_1}(C_s, C_t, p), f_{e_2}(C_s, C_t, p)\}, & \text{if } (C_s, C_t) \in (S_{e_1} \times T_{e_1}) \cap (S_{e_2} \times T_{e_2}) \\ f_{e_1}(C_s, C_t, p), & \text{if } (C_s, C_t) \in (S_{e_1} \times T_{e_1}) \setminus (S_{e_2} \times T_{e_2}) \\ f_{e_2}(C_s, C_t, p), & \text{if } (C_s, C_t) \in (S_{e_2} \times T_{e_2}) \setminus (S_{e_1} \times T_{e_1}) \\ \infty, & \text{otherwise.} \end{cases}$$

Again, we use $\infty$ as placeholder to signify combinations $(C_s, C_t, p)$ that are not valid in the sense that there exists no corresponding clique path, i.e. $\mathcal{P}_e(C_s, C_t, p)$ is empty.

**Generalized series composition 'G'** This is the most simple composition, as it neither introduces any new cycles in the dependency graph, nor does it require additional calculations. We simply have $S_e = S_{e_2}$, $T_e = T_{e_2}$, and for all $(C_s, C_t, p) \in S_e \times T_e \times \{0, 1\}$, we get

$$f_e(C_s, C_t, p) = f_{e_2}(C_s, C_t, p).$$

To summarize, the validity of constraints (11) can be checked in polynomial time (see [10]), and, if they are satisfied, separation of embedded OCC constraints is equivalent to separation of irreducible embedded OCC constraints. Together with the dynamic program explained above, we can then conclude the following.

**Theorem 6.13.** *Let $(G, \mathcal{V})$ be an instance of CPMCSP and $\tilde{x} \in \mathbb{R}^{|V(G)|}$ some vector. The time required to solve the separation problem for the class of embedded OCC constraints is polynomial in $|V(G)|$ and the number of maximal cliques in any subgraph $\overline{G}_{ij}$ of $\overline{G}$.*

We have now dealt with separation of embedded OCC constraints. Before moving on to our conclusions, we want to offer some closing remarks on the separation of non-embedded OCC constraints.

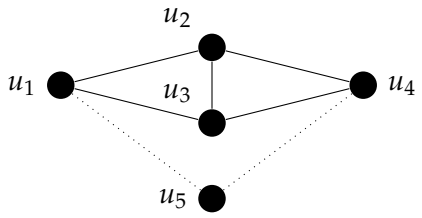



Figure 11: Separation of general OCC constraints in conflict graphs

In our algorithm, it is of crucial importance that the OCC constraints are embedded in the partition $\mathcal{V}$. Naturally, the question arises what can be done for non-embedded OCC constraints. Consider the conflict graph shown in Figure 11, where the five vertices $u_1 \in V_1, \dots, u_5 \in V_5$ are contained in different subsets $V_i$ of some given partition $\mathcal{V}$. Suppose there exists some clique cycle $\mathcal{C} = \{\{u_1\}, \{u_2, u_3\}, \{u_4\}, \dots, \{u_5\}, \dots\}$. The first three cliques $\{u_1\}, \{u_2, u_3\}, \{u_4\}$ induce in the dependency graph a $K_4$ that is missing a single edge. If the clique cycle does not pass through either of the subsets $V_2$, $V_3$ to connect back to $u_1$, then it induces a path connecting $V_1$ and $V_4$. This completes a $K_4$-minor in the dependency graph and, hence, the corresponding dependency graph is not series-parallel. From this observation, we can conclude two things: first, an odd clique hole must either be embedded in $\mathcal{V}$ or the cliques intersecting more than one subset in $\mathcal{V}$ induce a $K_4$-minor in $\mathcal{G}$. Second, there may exist OCC constraints that are not embedded but facet-defining. We therefore expect the separation of general odd-clique-hole constraints in conflict graphs with series-parallel dependency graphs to be possible within the same time complexity as the separation of embedded OCC constraints. The complexity of separating general OCC constraints remains unknown, but it stands to reason that this problem is more difficult as the 'locality' argument does not hold for OCC constraints.

# 7 Conclusions

In this work, we have introduced a new polynomial-time solvable special case of the clique problem with multiple-choice constraints – namely the one where the dependency graph is series-parallel (CPMCSP). On the one hand, it is a natural extension of the case where the dependency graph is a forest (CPMCF); see [10]. On the other hand, the study of CPMCSP was much more complicated as, in contrast to CPMCF, we could not rely on perfectness of the compatibility graph. We gave a dynamic programming algorithm to solve the optimization variant of the problem in $\mathcal{O}(|\mathcal{V}| \cdot |V_{\max}|^3)$ time, where $V_{\max} \in \mathcal{V}$ is a subset of maximum cardinality. Furthermore, we identified a subclass of instances of CPMCSP where the conflict graph is $h$-perfect. For the general stable set problem, we have derived a novel class of valid inequalities, which we call *odd-clique-cycle* (*OCC*) *constraints*, a generalization of the well-known odd-cycle constraints. Just like odd-cycle constraints, they induce facets if the clique cycle is a clique hole. However, an OCC constraint may be facet-defining even if its clique cycle is not a clique hole. A subset of these constraints, the *embedded OCC constraints*, can be separated efficiently via a dynamic programming algorithm if the maximal size of the subsets $V_i$ in the partition $\mathcal{V}$ is bounded. Altogether, this means that the new inequalities could be beneficial to use as cutting planes to reduce computation times in applications like scheduling problems, where CPMC often arises as a substructure (cf. Section 1). This might also work when the dependency graph is not series-parallel: By heuristically adding edges to the compatibility graph until the dependency graph becomes series-parallel, the feasible set of a CPMC instance is relaxed.

Future avenues of research include the transfer of our results to dependency graphs of

bounded treewidth greater than two. Among various real-world data sets, infrastructure networks have relatively low treewidth [24]. While absolute numbers are often still prohibitively large for algorithms that are exponential in the treewidth, algorithms for low-treewidth graphs may still be useful, as these networks 'tend to exhibit a tree-like fringe and a densely connected core' [24]. Based on this observation, *partial tree decompositions* have been suggested. The idea is to use algorithms for low-treewidth graphs on the fringe in order to reduce the network to its core, which only has a fraction of the size of the original network. For instance, the authors of [18] employ such considerations to reduce the size of large real-world gas networks by up to 76%. Furthermore, it would be interesting to study CPMC while assuming other notions of sparsity for the dependency graph, such as e.g. planarity. Finally, it remains an open question whether the separation problem for general, i.e. not necessarily embedded, OCC constraints can be done in polynomial time.

## References


[1] R. Aharoni, E. Berger, and R. Ziv. Independent systems of representatives in weighted graphs. *Combinatorica*, 27(3):253–267, 2007.

[2] R. Aharoni and P. Haxell. Hall's theorem for hypergraphs. *Journal of Graph Theory*, 35(2):83–88, 2000.

[3] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[4] H. L. Bodlaender and B. van Antwerpen-de Fluiter. Parallel algorithms for series parallel graphs and graphs with treewidth two. *Algorithmica*, 29(4):534–559, 2001.

[5] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, pages 1–74. Springer, 1999.

[6] R. Borndörfer. *Aspects of Set Packing, Partitioning, and Covering*. PhD thesis, Technische Universität Berlin, 1998.

[7] A. Bärmann, T. Gellermann, M. Merkert, and O. Schneider. Staircase compatibility and its applications in scheduling and piecewise linearization. *Discrete Optimization*, 29:111–132, 2018.

[8] A. Bärmann, P. Gemander, L. Hager, F. Nöth, and O. Schneider. EETTlib – Energy-Efficient Train Timetabling Library, 2021. Submitted; available under: `http://www.optimization-online.org/DB_HTML/2021/09/8597.html`.

[9] A. Bärmann, P. Gemander, A. Martin, and M. Merkert. On recognizing staircase compatibility, 2021. Submitted; available under: `http://www.optimization-online.org/DB_HTML/2020/12/8138.html`.

[10] A. Bärmann, P. Gemander, and M. Merkert. The clique problem with multiple-choice constraints under a cycle-free dependency graph. *Discrete Applied Mathematics*, 283:59–77, 2020.

[11] A. Bärmann, A. Martin, and O. Schneider. A comparison of performance metrics for balancing the power consumption of trains in a railway network by slight timetable adaptation. *Public Transport*, 9(1-2):95–113, 2017.

[12] A. Bärmann, A. Martin, and O. Schneider. Efficient formulations and decomposition approaches for power peak reduction in railway traffic via timetabling. *Transportation Science*, 55(3):747–767, 2021.

[13] A. Bärmann, A. Martin, and J. Staszek. A decomposition approach for integrated locomotive scheduling and driver rostering in rail freight transport, 2021. Submitted; available under: `http://www.optimization-online.org/DB_HTML/2021/08/8571.html`.

[14] V. Chvátal. On certain polytopes associated with graphs. *Journal of Combinatorial Theory, Series B*, 18(2):138–154, 1975.

[15] M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *4OR-Q J Oper Res*, 8(1):1–48, 2010.

[16] A. de Mier and M. Noy. On the maximum number of cycles in outerplanar and series–parallel graphs. *Graphs and Combinatorics*, 28(2):265–275, 2012.

[17] R. J. Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10(2):303–318, 1965.

[18] M. Gross, M. E. Pfetsch, L. Schewe, M. Schmidt, and M. Skutella. Algorithmic results for potential-based flows: Easy and hard cases. *Networks*, 73(3):306–324, 2019.

[19] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988.

[20] T. Grünert, S. Irnich, H.-J. Zimmermann, M. Schneider, and B. Wulfhorst. Finding all k-cliques in k-partite graphs, an application in textile engineering. *Computers & Operations Research*, 29(1):13–31, 2002.

[21] E. Hare, S. Hedetniemi, R. Laskar, K. Peters, and T. Wimer. Linear-time computability of combinatorial problems on generalized-series-parallel graphs. In *Discrete Algorithms and Complexity*, pages 437–457. Elsevier, 1987.

[22] A. D. King. Hitting all maximum cliques with a stable set using lopsided independent transversals. *Journal of Graph Theory*, 67(4):300–305, 2011.

[23] F. Liers and M. Merkert. Structural investigation of piecewise linearized network flow problems. *SIAM Journal on Optimization*, 26(4):2863–2886, 2016.

[24] S. Maniu, P. Senellart, and S. Jog. An Experimental Study of the Treewidth of Real-World Graph Data. In P. Barcelo and M. Calautti, editors, *22nd International Conference on Database Theory (ICDT 2019)*, volume 127 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:18, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[25] R. K. Martin, R. L. Rardin, and B. A. Campbell. Polyhedral characterization of discrete dynamic programming. *Operations Research*, 38(1):127–138, 1990.

[26] M. Mirghorbani and P. Krokhmal. On finding k-cliques in k-partite graphs. *Optimization Letters*, 7(6):1155–1165, 2013.

[27] M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5(1):199–215, 1973.

[28] C. Schwindt and J. Zimmermann, editors. *Handbook on Project Management and Scheduling (Vol. 1 + Vol. 2)*. Springer, 2015.

[29] J. A. Wald and C. J. Colbourn. Steiner trees, partial 2–trees, and minimum IFI networks. *Networks*, 13(2):159–167, 1983.

[30] T. Wimer and S. Hedetniemi. K-terminal recursive families of graphs. *Congressus Numerantium*, 63:161–176, 1988.

## Acknowledgements

This research was supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics – Data – Applications (ADA-Center) within the framework of "BAYERN DIGITAL II" (20-3410-2-9-8). Part of this work was done while M. Merkert was affiliated with Otto-von-Guericke-Universität Magdeburg. This author thanks the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for their support within GRK 2297 MathCoRe. Further, this research has been supported by funding of the Bavarian State Government through the Energie Campus Nürnberg. A. Wiertz thanks the DFG for their support within the project B09 in CRC TRR 154. Finally, this research was partly funded by the Deutsche Akademische Austauschdienst (DAAD) with means from the Bundesministerium für Bildung und Forschung (BMBF) through a Gastdozentur at the Chair of EDOM in the Department of Mathematics of the Friedrich-Alexander-Universität Erlangen-Nürnberg (Projekt-ID 57425212) while F. Zaragoza Martínez was enjoying a sabbatical leave from Universidad Autónoma Metropolitana Azcapotzalco. This author also acknowledges the support of the National Council of Science and Technology (Conacyt) and its National System of Researchers (SNI).